Aalto University
School of Science
Degree Programme of Computer Science and Engineering

Orestis Kostakis

Analyzing and comparing arrangements of temporal intervals

Master's thesis

Espoo, November 5, 2011

Supervisor: Prof. Pekka Orponen

Instructor: Panagiotis Papapetrou, PhD

Aalto University
School of Science

Aalto University School of Science Degree Programme of Computer Science and Engineering	ABSTRACT OF MASTER'S THESIS

Author:	Orestis Kostakis				
Title:	Analyzing and comparing arrangements of temporal intervals				
Number of pages:	vi + 38	Date:	November 5, 2011	Language:	English
Professorship:	Theoretical Computer Science		Code:		T-79
Supervisor:	Prof. Pekka Orponen				
Instructor:	Panagiotis Papapetrou, PhD				

Abstract:

This thesis focuses on comparing and analyzing arrangements of temporal intervals. Such arrangements are sets of concurrent events that are not instantaneous, but are characterized by duration.

We study two major problems. The first problem is comparing arrangements of event-intervals and acquiring their distance. To the best of our knowledge, we are the first to formally define the problem. Furthermore, we present three polynomial-time distance functions which we study and benchmark through rigorous experimentation. The proposed methods were tested on three datasets: American Sign Language utterances, sensor data and Hepatitis patient data. In addition, we provide a linear-time lower bound for one of the distance measures. The distance measures can be applied to event-interval sequences, too. In this case, neither the event-interval durations nor the absolute time values are considered.

The second problem which we study is finding the longest common sub-pattern (LCSP) of arrangements of temporal intervals. We prove hardness results for the problem and devise an exact algorithm for computing the LCSP of pairs of arrangements.

Keywords: Arrangements of temporal intervals, Event-interval sequences, Distance measure, Longest Common Sub-pattern (LCSP), Lower bound.

Preface

This thesis is submitted in partial fulfillment of the requirements for a Computer Science and Engineering, Master's Programme on Foundations of Advanced Computing. It consists of the formal documentation of work performed by the author during the period January - May 2011, in close collaboration with the "Data Mining: Theory and Applications" group. This thesis has been supervised by Prof. Pekka Orponen, while Dr. Panagiotis Papapetrou has been the instructor.

The thesis topic resulted from an informal conversation with Dr. Papapetrou in Autumn 2010. The following months involved successful efforts to solve a series of interesting problems. The ideas and the results presented in this thesis have already received positive feedback from the academic community. Different parts have already been presented at two peer-reviewed scientific conferences [26, 25].

Acknowledgments

The submission of this master thesis implies the completion of my studies and work at TKK/ Aalto University. After two creative years, I leave full of great memories. During my presence at the Department of Information and Computer Science, I had the chance to meet and collaborate with outstanding colleagues. However, all this would not have been possible if I had not joined the Combinatorial Algorithms and Computation group as a Summer Trainee in 2009. Therefore, I feel it is imperative to sincerely thank Professor Pekka Orponen, for being not only a remarkable supervisor, but also for accepting me as an equal member of the group from the first day.

In similar manner, I would like to thank the "Data Mining: Theory and Applications" group for the collaboration related to my Master's Thesis. In particular, I would like to thank Dr. Panagiotis Papapetrou for being the instructor of this thesis. Additionally, special thanks go to Dr. Jaakko Hollmén.

Moreover, I thank all the colleagues and peers with whom I had the opportunity to work as part of the Future Internet project. The experience gained is truly invaluable. Last but not least, to all my teachers and my classmates (especially the FAdCo class of 2011: Nguyen Chi Mai, Rehan Abdul Aziz and Tao Sun) for contributing to the success of my MSc studies.

Otaniemi, November 5, 2011

Orestis N. Kostakis

Contents

Abbreviations

ASL	American Sign Language
BFS	Breadth-First Search
DFS	Depth-First Search
DTW	Dynamic Time Warping
EIS	Event-interval Sequence
ED	Euclidean Distance
LCSP	Longest Common Sub-pattern
TIS	Temporal Interval Sequence

List of Figures

List of Tables

Chapter 1

Introduction

Temporal interval sequences (or event-interval sequences) allow the representation of all possible series of temporal events. 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 fields such as geoinformatics [43], cognitive science [6], linguistic analysis [7], music research [38] and medicine [24]. Essentially, temporal interval sequences can be encoded as a collection of labeled events accompanied by their start and end time values. An example of a sequence of five event-intervals is shown in Figure 1.1.

So far, the work related to event-interval sequences has focused on the aspect of knowledge discovery. Sequences are mined in order to discover patterns and association rules that might be of interest to the researchers [19, 3]. However, surprisingly no work has been performed on comparing event-interval sequences. Often, it is required to assess the similarity of pairs of sequences or to infer about the existence of a single sequence in a database. Furthermore, once a framework for comparing temporal interval sequences has been defined, it would enable many types of classification and clustering algorithms and facilitate the implementation of solutions such as recommender systems, phylogenies and assistive applications. Existing similarity measures focus on either symbolic sequences or time series and it is not possible to directly apply them on event-interval sequences (this is further discussed in Section 4.1). To the best of our knowledge, there has been yet no robust distance or similarity measure for comparing sequences of event-intervals.

During the course of this thesis, we were able to provide several contributions of varying importance. In the following chapters, we formulate the problem of comparing sequences of event-intervals and show that solving the problem by directly mapping it to string matching fails to capture any temporal dependencies between the events. Furthermore, we propose three distance measures to solve this problem: the first maps the event-interval sequence to a sequence of vectors that corresponds to a multi-dimensional time series. The second measure enumerates all event-interval relations within each sequence and reduces the problem to that of comparing matrices. Finally, the third distance measure attempts to find corresponding or equivalent event-intervals and employs maximum bipartite matching. Moreover, we propose a lower bounding technique for the third method that achieves significant speed-ups during similarity search. In order to acquire an insight into the behavior of the three proposed methods, we present an extensive experimental evaluation on real datasets from three different domains. The methods are benchmarked with respect to their robustness to noise, nearest-neighbor classification accuracy, and scalability. In addition, we study the pruning power and tightness of the proposed lower bound. Furthermore, we study the problem of finding the longest common sub-arrangement (or sub-pattern) of pairs of temporal interval arrangements. Lastly, we define the problem of finding the longest common sub-pattern (LCSP), for which we prove hardness properties and provide an exact algorithm.

This thesis is arranged as follows: existing related work is presented in Chapter 2 and in Chapter 3 we provide the required background, which includes a description of the used

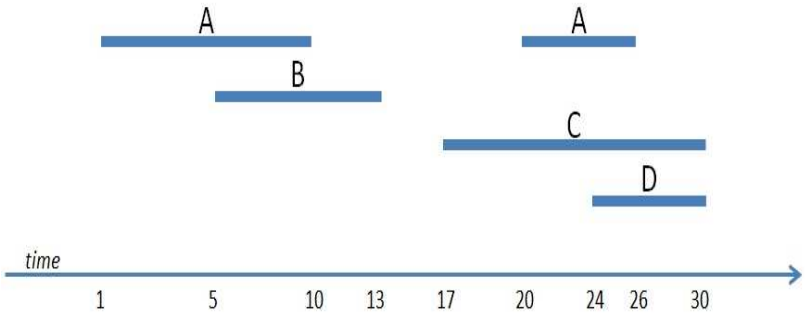


Figure 1.1: An example of a sequence of event-intervals.

notation and terminology. In Chapter 4 we formally define the proposed distance functions, while we devote Chapter 5 to the problem of the longest common sub-pattern. The experimental setup, the results and our conclusions are presented in Chapter 6. Finally, in Chapter 7 we provide a summary of this thesis and several directions for future work.

Chapter 2

Related Work

Existing work on interval-based sequences has so far been focusing merely on frequent pattern and association rule mining.

Several approaches [30, 46] consider discovering frequent intervals in databases, where intervals appear sequentially and are not labeled, while others [13] consider temporally annotated sequential patterns where transitions from one event to another have a time duration. A graph-based approach [16] represents each temporal pattern by a graph considering only two types of relations between events (*follow* and *overlap*). An approach for mining sequences of interval-based events in a database is discussed by Kam and Fu [19], however it is limited to certain forms of patterns.

A generalized interval-based framework [28] improves support counting techniques for mining interval-based episodes; nonetheless, no temporal relations are considered between events. Apriori-based techniques [1, 10, 14, 15, 32] for finding temporal patterns and association rules on interval-based event sequences have been proposed, some [15] also applying interestingness measures to evaluate the significance of the findings.

BFS-based and DFS-based approaches [48, 40, 39, 41] apply efficient pruning techniques, thus reducing the inherent exponential complexity of the mining problem, while a non-ambiguous event-interval representation is defined in [49] that considers start and end points of event sequences and converts them to a sequential representation. In addition, there has been some recent work on mining semi-partial orders of time intervals [35].

Moreover, in Ale et al. [3], 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 interval. Finally, Lu et al. [31] study inter-transaction association rules by merging all itemsets within a sliding time window inside a transaction.

Recent work on *margin-closed* patterns [34, 35] focuses on significantly reducing the number of reported patterns by favoring longer patterns and suppressing shorter patterns with similar frequencies. The extracted set of margin-closed patterns may include a significantly smaller set of patterns compared to the set of closed patterns while retaining the most important information about the data. A unifying view of temporal concepts and data models has been formulated in [33] 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.

Chapter 3

Background

In this Chapter we provide the necessary background related to our work. In Section 3.1 we define the used terminology and notation, while in Section 3.2, for completeness, we describe the Dynamic Time Warping algorithm used for time series matching.

3.1 Event-Interval Sequences

Given an alphabet σ of event labels, a triple $S_i = (E_i, t^i_{start}, t^i_{end})$ is called an *event-interval*, where $E_i \in \sigma$ and t^i_{start}, t^i_{end} denote the start and end time of S_i, respectively. E_i is in some cases equivalently denoted as E_{Si} to achieve clarity. The values of t^i_{start} and t^i_{end} can be real numbers. In general, $t^i_{start} \leq t^i_{end}$; the equality holds when the event is instantaneous. Let $\mathcal{S}$=$\{S_1, \ldots, S_n\}$ be a sequence of events occurring at time intervals. $\mathcal{S}$ is called an *event interval sequence*, or *e-sequence* [40, 41]. In the current context, event-intervals are equivalent to temporal intervals, so e-sequences are equivalent to temporal interval sequences. 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, alphabetical ordering is applied based on the event-interval labels. An example of an e-sequence is shown in Figure 1.1, and it corresponds to:

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

In several cases, the duration or the absolute time values of the start and end points of the event-intervals are not important. Hence, an additional term, that of *temporal interval arrangements* or simply *arrangements* [40, 41], can be used to describe sequences of temporal intervals. In this case, the important information is in the temporal relations between the intervals. Formally, an arrangement $\mathcal{A} = \{\mathcal{E}, \mathcal{R}\}$ of n events consists of a sequence of event labels $\mathcal{E}$ that contains the labels of all temporal intervals included in $\mathcal{A}$, with $|\mathcal{E}| = n$, and a set of relations $\mathcal{R} = \{R(E_1, E_2), R(E_1, E_3), \ldots, R(E_{n-1}, E_n)\}$, where each $R(E_i, E_j)$ denotes the temporal relation between (E_i, E_j), for $i = 1, \ldots, n-1$ and $j = i+1, \ldots, n$, with $R(E_i, E_j) \in \mathcal{I}$. $\mathcal{I} = \{r_1, \ldots, r_{|\mathcal{I}|}\}$ is the set of all legal temporal relations that can exist between any pair of events. We use the term $R_l(E_i, E_j)$, called *labeled relation*, to denote the combination of both the interval relation $R(E_i, E_j)$ of a pair of intervals and the ordered pair of interval labels (E_i, E_j). Then, for a pair of labeled relations, $R_l(E_i, E_j) = R_l(E_k, E_l)$ holds if and only if $R(E_i, E_j) = R(E_k, E_l)$, $E_i = E_k$ and $E_j = E_l$. The definition of the labeled relation is equivalently applied to pairs of event-intervals in e-sequences. Finally, the size of an arrangement $\mathcal{A} = \{\mathcal{E}, \mathcal{R}\}$ is equal to $|\mathcal{E}|$. Although $|\mathcal{R}| = \frac{|\mathcal{E}|(|\mathcal{E}|-1)}{2}$, it is possible to reconstruct $\mathcal{R}$ from a event-interval sequence of size $|\mathcal{E}|$ with nominal time values.

Based on Allen's model for temporal intervals and their relations [5, 4, 40, 41] we consider the following seven relations for two event-intervals A and B. This is also presented in Figure 3.1.

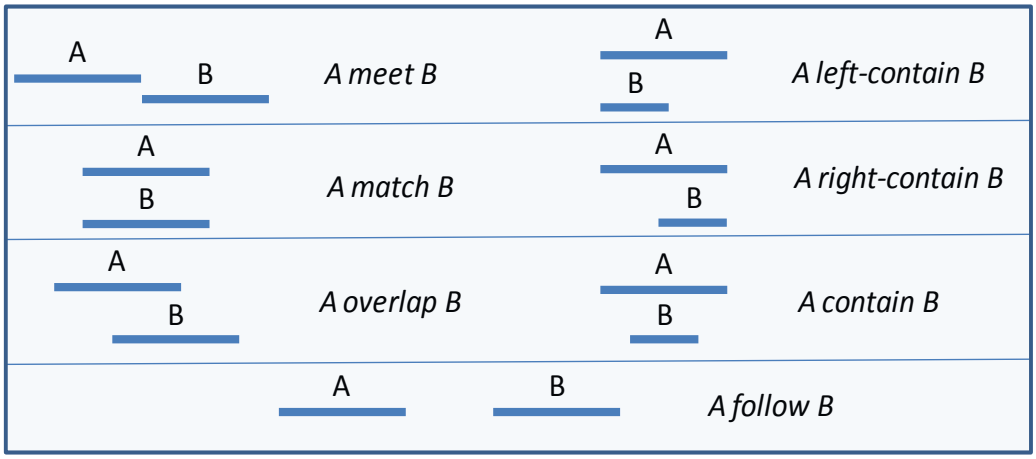


Figure 3.1: The seven temporal relations between two event-intervals that are considered in this thesis.

- **follow**(A,B) denotes the case where A occurs before B; A ends before B begins, $t^A_{end} \leq t^B_{start}$.
- **meet**(A,B) denotes the case where A meets B; B starts at the same time that A ends, $t^A_{end} = t^B_{start}$.
- **match**(A,B) denotes the case where A matches B; A starts and ends at the same time as B, $t^A_{start} = t^B_{start}$ and $t^A_{end} = t^B_{end}$.
- **overlap**(A,B) denotes the case where A and B have overlapping parts; A starts before B and B starts before A has ended, $t^A_{start} < t^B_{start}$ and $t^A_{end} > t^B_{start}$.
- **contain**(A,B) denotes the case where A contains B; A starts before B starts and A ends after B has ended, $t^A_{start} < t^B_{start}$ and $t^A_{end} > t^B_{end}$.
- **left contain**(A,B) denotes the case where A left-contains B; A and B start simultaneously and A ends after B, $t^A_{start} = t^B_{start}$ and $t^A_{end} > t^B_{end}$.
- **right contain**(A,B) denotes the case where A right-contains B; A starts before B, but end simultaneously, $t^A_{start} < t^B_{start}$ and $t^A_{end} = t^B_{end}$.

We do not consider symmetric relations, i.e. B is contained by A, since these can be expressed by the above and thus are redundant. Hence, $\mathcal{I} = \{meet, match, overlap, contain, left\ contain, right\ contain, follow\}$ and $|\mathcal{I}| = 7$.

Furthermore, as discussed in Papapetrou et al. [41] there may exist ambiguities between the aforementioned relations due to noise in the data; for simplicity, we do not consider this in our work.

Using the above definitions, the problem studied in this thesis is formulated as follows:

Problem 3.1. ***(E-sequence Distance)*** *Given two e-sequences $\mathcal{S}$ and $\mathcal{T}$, define a distance measure D, such that $\forall \mathcal{S}, \mathcal{T}$ it holds: $D(\mathcal{S}, \mathcal{T}) \geq 0$, $D(\mathcal{S}, \mathcal{S}) = 0$ and $D(\mathcal{S}, \mathcal{T}) = D(\mathcal{T}, \mathcal{S})$. The degree to which the two e-sequences differ should be reflected in the value of $D(\mathcal{S}, \mathcal{T})$.*

Obviously, if the last sentence was not present in the problem definition, then $D(\mathcal{S}, \mathcal{T}) = 0$ would satisfy all the conditions, but would be of no practical use. In addition, the scores returned by the distance measure should be in accordance with the knowledge obtained from domain experts.

3.2 Dynamic Time Warping

In this Section we provide a description of the *Dynamic Time Warping* (DTW) algorithm [8, 27]. DTW takes as input two *time series* [9] and returns their distance score. As the name suggests, the algorithm is based on the notion of dynamic programming. DTW is a very popular choice for full-sequence matching, since it offers a series of advantages over the *Euclidean distance* (ED).

Time series are sequences of real numbers representing the measurements of a real variable, sampled at equal time intervals. They can represent the fluctuation in the value of variables such as stock prices, ECG measurements and temperature. Time series are a valuable representation of data in the cases where the order of the values are of importance. Driven by the need to index and retrieve similar time series, scientists have devised DTW.

The aim of the DTW algorithm is to compute the distance between pairs of time series. The algorithm implicitly attempts to identify common regions or patterns between the sequences. One of the main characteristics of DTW is the possibility to allow non-linear matchings of the time points in the two time series. An example of that can be seen in Figure 3.2. Matching multiple points of one time series to a single point of the other translates to adding stutter in the specific point of the second time series. Equivalently, it can represent a stretch of the time series along the time axis. This allows to overcome the problem of the Euclidean distance that requires the time series to be of equal length. The algorithm computes all valid matchings and returns the one of minimum cost. The cost of a matching is equal to the sum of the difference in the values of all matched pairs of points. The general form of DTW is depicted in Algorithm 1. The Euclidean distance is equivalent to the case of DTW where the only allowed dynamic path is that of the diagonal. Thus, DTW can be seen as a generalization of the Euclidean distance.

To compute the DTW distance of two time series of length n and m, time $O(nm)$ is required, since $n \cdot m$ prefix matching costs have to be computed. Various approaches to speed up the computation have been proposed. These can be divided into two main categories. The first consists of techniques such as the Sakoe-Chiba band [44] and the Itakura parallelogram [17], which, by aiming to prevent pathological warpings, impose a constraint on the valid

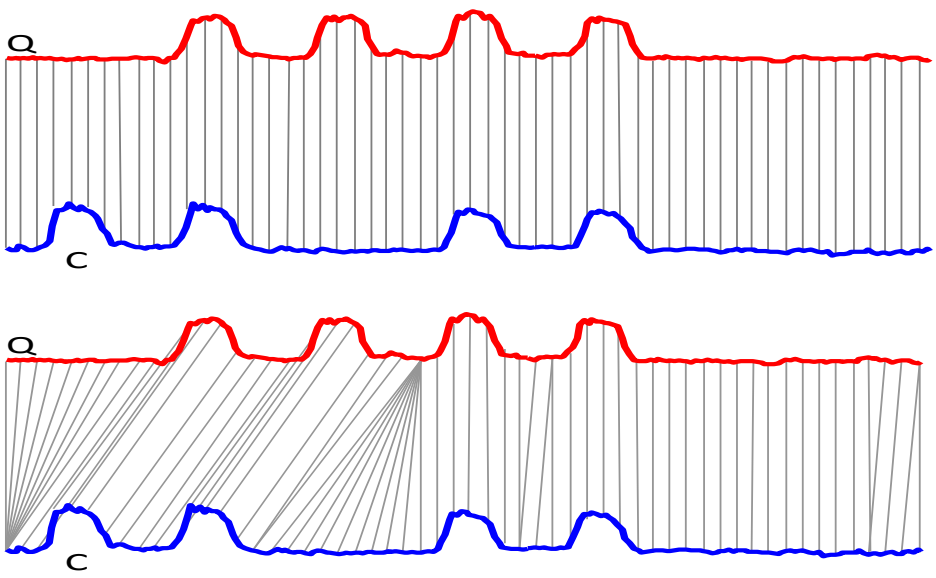


Figure 3.2: Comparison of the matchings allowed in Euclidean distance (top) and DTW (bottom). ED allows only 1-1 matching of the points that are in the same position. In contrast, DTW allows any time point of any sequence to be matched to multiple consecutive time points in the other. Figure taken from [22].

dynamic paths. The second category consists of constant- or linear-time lower bounds. Lower bounding techniques approximate the real distance value by using a subset of the time series' values. Several lower bounding techniques have been proposed [22, 23, 50] for use in time series databases.

The DTW distance does not satisfy the triangular inequality and thus, is not a metric. Nevertheless, its use is very broad since it provides the basis for solving additional problems such as that of subsequence matching, e.g., with algorithms such as `SPRING` [45].

Algorithm 1 Dynamic Time Warping

Input: Two time series $X = x_1, \cdots, x_n$ and $Y = y_1, \cdots, y_m$.
Output: $DTW(X, Y)$

Declare `int` $DTW[0..n, 0..m]$;
Declare `int` i, j, $cost$;

/* Initialization */
$DTW[0, 0] = 0$;
for $i = 1$ to n **do**
  $DTW[i, 0] = \infty$;
end for
for $i = 1$ to m **do**
  $DTW[0, i] = \infty$;
end for

/* Main Procedure */
for $i = 1$ to n **do**
  for $j = 1$ to m **do**
    $cost = |x_i - y_j|$;
    $DTW[i, j] = cost + \min(DTW[i-1, j],$
                $DTW[i, j-1],$
                $DTW[i-1, j-1])$;
  end for
end for
return $DTW[n, m]$;

Chapter 4

Distance functions

In this section we propose three distance functions to compare event-interval sequences. The first is based on mapping event-interval sequences to vectors. In Section 4.1 we define the vector-based DTW distance function. The second distance function focuses on the set of interval relations within each e-sequence. The distance of a pair of e-sequences is derived by assigning equal weight to each interval relation. This becomes possible by mapping each e-sequence to a Relation Matrix and, consequently, comparing the two matrices. The *Relation Matrix* distance function is defined in Section 4.2. Finally, the third distance function, `Artemis`, focuses on the interval relations, too, but the distance score is inferred after finding corresponding intervals in the pair of e-sequences. `Artemis` computes relation sets and applies maximum bipartite matching; the details are described in Section 4.3. Additionally, in Section 4.4 we provide a linear-time lower-bounding technique for `Artemis`.

In the following sections, it will become apparent that neither the duration of temporal events nor the time separating them is taken into account. As a result, scaling the temporal values of an e-sequence does not affect the result when compared to another e-sequence. Most importantly, the methods are directly applicable to temporal interval arrangements.

4.1 The Vector-based DTW Distance

An intuitive approach to compare event-interval sequences would be to map them to strings, by projecting the start and end points onto the time axis, and then apply string matching or string edit algorithms, such as computing the Levenshtein distance [29]. Due to the fact that events can start or end concurrently, it is impossible to apply any ordering on the symbols that would not create ambiguities [26]. Thus, one has to resort to the use of vectors to encode all the information for a specific time-point.

Our first method employs a vector-based representation of e-sequences. For each e-sequence, a set of vectors is defined (technically, it is a multi-set), where each vector indicates which and how many event-intervals are active at specific time points in the e-sequence. The selected time points are all the instances in which an event-interval begins or ends.

Definition 4.1. ***(Event Vector)*** *Given an e-sequence $\mathcal{S}$ defined over an alphabet σ, an event vector $V^t = (V_1^t, \ldots, V_{|\sigma|}^t)$ consists of integer values, where each V_i^t records the number of occurrences of events with label $i \in \sigma$, at time stamp t in $\mathcal{S}$.*

Hence, an e-sequence $\mathcal{S}$ can be mapped to a sequence of event vectors $\mathcal{V}_\mathcal{S} = \{V^{t_0}, V^{t_1}, \ldots, V^{t_m}\}$. The set of time stamps $\{t_1, \ldots, t_m\}$ includes all time points in $\mathcal{S}$ where the status of at least one event-interval changes, i.e., an event-interval starts or ends. V^{t_0} is the null vector which denotes the initial condition at t_0 where no event takes place. In addition, V^{t_m} is always the null vector, since all events have ended. An example of mapping an e-sequence to event vectors, can be seen in Figure 4.1.

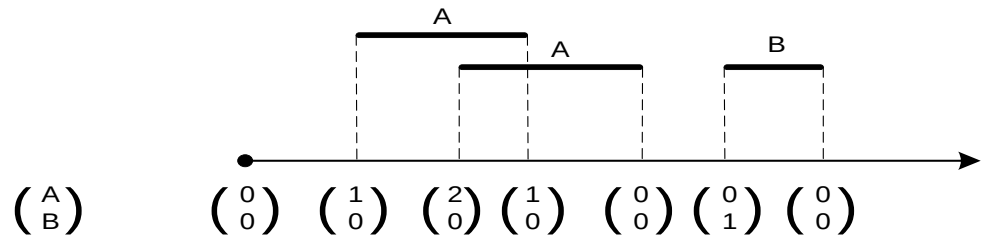


Figure 4.1: Mapping of arrangements or e-sequences to event vectors. The above arrangement can be encoded as a sequence of 7 event vectors. Once the vectors are computed, multi-dimensional DTW can be applied.

Given two e-sequences $\mathcal{S}$ and $\mathcal{T}$, their vectors can also be seen as $|\sigma|$-dimensional time series; thus, their distance can be computed using Dynamic Time Warping. In order to avoid assigning unequal weight between any combinations of event-intervals, the vectors are compared using the $\mathcal{L}_1$ norm.

Complexity. The sets of event vectors $\mathcal{V}_\mathcal{S}$ and $\mathcal{V}_\mathcal{T}$ can be computed in $O(|\mathcal{V}_\mathcal{S}||\sigma| + |\mathcal{V}_\mathcal{T}||\sigma|)$ time. The time complexity of this DTW computation is $O(|\mathcal{V}_\mathcal{S}||\mathcal{V}_\mathcal{T}||\sigma|)$. Significant speedup in similarity search under DTW can be achieved using the LB_Keogh lower bound extension for multidimensional time series [47].

Despite the simplicity of this method, it only takes into account the temporal ordering of the event-intervals but does not explicitly consider any temporal relations. This may result in exactly matching two different e-sequences by mapping them to the same set of event vectors; such an example is depicted in Figure 4.2.

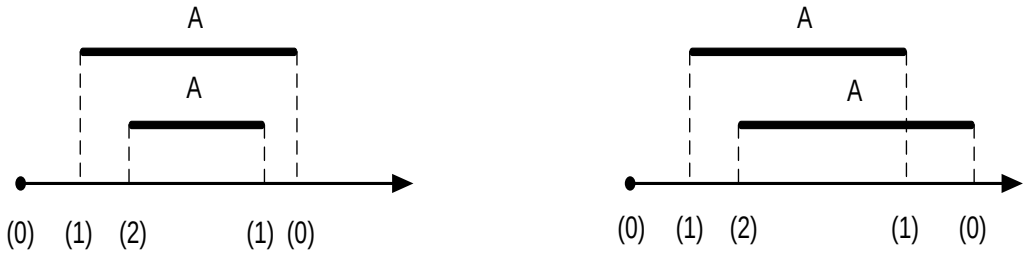


Figure 4.2: Two arrangements for which DTW yields a zero distance score, although they represent different situations.

4.2 Relation Matrix

4.2.1 Creation of Relation Matrix

In this section we define a method to compare e-sequences by enumerating all their interval relations and assigning to each one equal weight. Focus is given only on the relations among the event-intervals, disregarding absolute time values. The steps involved in this computation include: (1) discarding the time stamps and representing each of the two e-sequences as an arrangement, (2) mapping each arrangement to a *relation matrix*. The two matrices are then compared to derive the distance score.

To define a relation matrix for an e-sequence $\mathcal{S}$ we should first map the e-sequence to an arrangement $\mathcal{A} = \{\mathcal{E}, \mathcal{R}\}$. The relation matrix $M_\mathcal{A}$ of $\mathcal{A}$ is an $|\mathcal{I}| \times |\sigma|^2$ integer-valued matrix

that keeps track of the count of all $|\mathcal{I}|$ types of temporal relation pairs that may occur in the arrangement. By defining the ordered set of event-label pairs $\mathcal{P} = \{\sigma \times \sigma\}$ and P_j the jth pair in the set, we can easily define the Relation matrix as follows.

Definition 4.2. *Given an arrangement $\mathcal{A}$, the corresponding relation matrix $M_{\mathcal{A}}$ is defined as follows:*

$$M_{\mathcal{A}}(i,j) = |\{R_l(E_k,E_l)|R(E_k,E_l) = r_i \text{ and } (E_k,E_l) = P_j\}|, \tag{4.1}$$

$\forall r_i \in \mathcal{I},\ j = 1,\ldots,|\sigma|^2,\ k = 1,\ldots,|\mathcal{E}|-1,\ l = k+1,\ldots,|\mathcal{E}|.$

Rows of $M_{\mathcal{A}}$ correspond to relations among event-intervals (as defined in $\mathcal{I}$) and columns correspond to pairs of interval labels in σ. The value of each cell is the number of times the labeled relation occurs in $\mathcal{A}$. For example, $M_{\mathcal{A}}(1,1)$ denotes the number of times relation r_1 appears between (σ_1,σ_1) in $\mathcal{A}$ (where σ_1 the first element of the label alphabet σ).

For any arrangement $\mathcal{A}$ of size m, it holds that

$$\sum_{i=1}^{|\mathcal{I}|}\sum_{j=1}^{|\sigma|^2} M_{\mathcal{A}}(i,j) = \frac{m(m-1)}{2}. \tag{4.2}$$

To better illustrate this mapping, the example in Table 4.1 demonstrates the relation matrix of the arrangement shown in Figure 4.3.

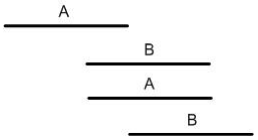


Figure 4.3: An arrangement with alphabet $\sigma = \{A, B\}$. Its corresponding relation matrix is shown in Table 4.1.

relation	(A,A)	(A,B)	(B,A)	(B,B)
meet	0	1	0	0
match	0	0	1	0
overlap	1	2	0	1
contain	0	0	0	0
left-contain	0	0	0	0
right-contain	0	0	0	0
follow	0	0	0	0

Table 4.1: The relation matrix $M_{\mathcal{A}}$ of arrangement $\mathcal{A}$ shown in Figure 4.3.

4.2.2 Comparison of Relation Matrices

Suppose we would like to compare e-sequences $\mathcal{S}$ and $\mathcal{T}$. We should first express these e-sequences with respect to their arrangement representation (say, $\mathcal{A}$ and $\mathcal{B}$, respectively), i.e., ignoring the event-interval durations and considering only the temporal relations between the events. Then each arrangement will be mapped to its corresponding relation matrix representation; thus creating matrices $M_{\mathcal{A}}$ and $M_{\mathcal{B}}$. Consequently, the problem of comparing the original e-sequences is mapped to the problem of comparing their relation matrices.

Definition 4.3. *We define the following generalized arrangement distance function:*

$$\delta_p(\mathcal{A},\mathcal{B}) = \left(\sum_{i=1}^{|\mathcal{I}|}\sum_{j=1}^{|\sigma|^2} |M_{\mathcal{A}}(i,j) - M_{\mathcal{B}}(i,j)|^p\right)^{\frac{1}{p}}, \quad p \in \mathbb{N}^* \tag{4.3}$$

The Manhattan Distance

For $p = 1$, Equation 4.3 yields the entry-wise *Manhattan* distance between $M_{\mathcal{A}}$ and $M_{\mathcal{B}}$. Using this distance, it is possible that the comparison of two equal sized arrangements yields a higher score than the comparison of one of them with a significantly smaller arrangement. For example, consider three arrangements $\mathcal{A}, \mathcal{B}, \mathcal{C}$ with 5, 10, and 10 event-intervals, respectively. Effectively, this means that $\mathcal{A}$ contains 10 possible temporal relations, while $\mathcal{B}$ and $\mathcal{C}$ contain 45. Now suppose that $\mathcal{A}$ and $\mathcal{B}$ agree on 8 relations, whereas $\mathcal{B}$ and $\mathcal{C}$ agree on 25 relations. Then, $\delta_1(\mathcal{A},\mathcal{B}) = 2 + 37 = 39$, since they differ on the remaining 2 relations of $\mathcal{A}$ and the remaining 37 relations of $\mathcal{B}$. Similarly, $\delta_1(\mathcal{B},\mathcal{C}) = 20+20 = 40$. This suggests that $\mathcal{B}$ is more similar to $\mathcal{A}$ than $\mathcal{C}$.

By a more detailed examination, however, one may easily notice that the total number of relations that may exist between $\mathcal{A}$ and $\mathcal{B}$ is 55, while 90 relations may exist between $\mathcal{B}$ and $\mathcal{C}$. This means that the former pair of arrangements agree on a smaller fraction of relations (i.e., $\frac{2\times 8}{55} \approx 0.28$) than the later pair (i.e., $\frac{2\times 25}{90} \approx 0.56$). Yet, by the Manhattan Distance the first pair appears to be more similar compared to the second. If the Manhattan distance were to be adopted, this anomaly would propagate into further procedures, such as clustering, giving incorrect results.

The Normalized Manhattan Distance

For the reason mentioned, we propose the following normalized version of δ_1:

Definition 4.4. *Given arrangements $\mathcal{A}$ and $\mathcal{B}$, the normalized Manhattan distance is defined as follows:*

$$\begin{aligned}\delta_{norm}(\mathcal{A},\mathcal{B}) &= \frac{\sum_i \sum_j |M_{\mathcal{A}}(i,j) - M_{\mathcal{B}}(i,j)|}{\frac{|\mathcal{A}|(|\mathcal{A}|-1)}{2} + \frac{|\mathcal{B}|(|\mathcal{B}|-1)}{2}} \\ &= 2 \times \frac{\sum_i \sum_j |M_{\mathcal{A}}(i,j) - M_{\mathcal{B}}(i,j)|}{|\mathcal{A}|(|\mathcal{A}|-1) + |\mathcal{B}|(|\mathcal{B}|-1)}\end{aligned} \tag{4.4}$$

For any pair of arrangements $\mathcal{A}$ and $\mathcal{B}$ the following three properties hold:

$$0 \leq \delta_{norm}(\mathcal{A},\mathcal{B}) \leq 1 \tag{4.5}$$

$$\delta_{norm}(\mathcal{A},\mathcal{A}) = 0 \tag{4.6}$$

$$\delta_{norm}(\mathcal{A},\mathcal{A}_\emptyset) = 1 \tag{4.7}$$

where $\mathcal{A}_\emptyset$ corresponds to the arrangement of the null e-sequence, i.e., an e-sequence without any event-intervals.

The Frobenius Norm

For $p = 2$, the distance expressed by Equation 4.3 is equal to the *Frobenius norm* of $(M_{\mathcal{A}} - M_{\mathcal{B}})$:

$$\delta_2(\mathcal{A},\mathcal{B}) = \sqrt{\sum_{i=1}^{|\mathcal{I}|}\sum_{j=1}^{|\sigma|^2} |M_{\mathcal{A}}(i,j) - M_{\mathcal{B}}(i,j)|^2} \tag{4.8}$$

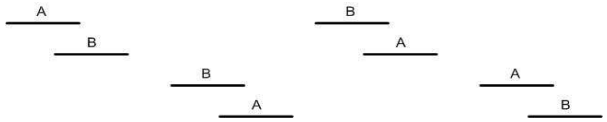
4.2.3 Properties

Proposition 4.1. *Distance function δ_p is not a metric for any $p \in \mathbb{N}$.*

Proof. δ_p violates the identity of indiscernibles (also known as Leibniz's law). There exist arrangements $\mathcal{A} \neq \mathcal{B}$ such that $\delta_p(\mathcal{A}, \mathcal{B}) = 0$. An example of such pair is shown in Figure 4.4. Clearly, $M_\mathcal{A}$ and $M_\mathcal{B}$ are identical even though they correspond to different arrangements. □

Proposition 4.2. *There exists an infinite number of pairs of arrangements that, although they are different, have the same Relation Matrix representation and thus, yield a zero δ_p distance score.*

Proof. By appending any arrangement (combination) of event-intervals to the arrangements mentioned in Proposition 4.1, we create pairs of arrangements that continue to have the same Relation Matrix representation. This holds because the induced relations between the existing arrangement and the appended are only of type *follow* and between event-intervals of the same label pair combination. The relations corresponding to the appended arrangement are identical in both cases. So, an infinite number of arrangements(of arbitrary size and label combinations) can be appended, thus, there exists an infinite number of pairs of arrangements $\mathcal{A}$, $\mathcal{B}$ such that $\delta_p(\mathcal{A}, \mathcal{B}) = 0$. □



(a) Arrangement $\mathcal{A}$ (b) Arrangement $\mathcal{B}$

Figure 4.4: Two arrangements for which δ_p violates the identity of indiscernibles.

Complexity. In the case where basic arithmetic operations can be performed in constant time, i.e., addition, multiplication, and square root need $O(1)$ time units, the time needed to compare two arrangements $\mathcal{A}$ and $\mathcal{B}$ of sizes n and m respectively is $O(n^2 + m^2 + |\sigma|^2)$, assuming that both arrangements are defined over the same alphabet σ.

4.3 `Artemis`: A Bipartite-based Matching Distance

In this section we describe an alternative method, called `Artemis`—shorthand for Assessing coRrespondence of Temporal Events Measure for Interval Sequences—which is based on determining correspondence between pairs of event-intervals to infer the overall similarity of e-sequences. Given a pair of event-intervals belonging to two different e-sequences, correspondence is determined by the fraction of common interval relations. The overall distance score is derived from the sum of pairwise scores using the Hungarian algorithm (also known as Kuhn-Munkres' algorithm) [37]. `Artemis` consists of two main steps: (a) the mapping step and (b) the matching step.

The mapping step. The first step of `Artemis` 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:

- $r_{left}(S_i) = \{R_l(S_j, S_i) | 1 \leq j < i\}$, which contains the labeled temporal relations of S_i with all event-intervals preceding S_i in $\mathcal{S}$.
- $r_{right}(S_i) = \{R_l(S_i, S_j) | i < j \leq |S|\}$, which contains the labeled temporal relations of S_i with all event-intervals succeeding S_i in $\mathcal{S}$, and

- $r_{\varnothing}(S_i) = \{R_l(\varnothing, S_i)\}$, 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: $r_{\varnothing left}(S_i) = r_{left}(S_i) \cup r_{\varnothing}(S_i)$.

Definition 4.5. ***(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:*

$$r(S_i) = r_{left}(S_i) \cup r_{right}(S_i) \cup r_{\varnothing}(S_i). \tag{4.9}$$

Note that $\varnothing$ is introduced so that event-interval labels are also taken into account: 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. Another important note is that when enumerating the three sets of relations, the actual labels and the order of the pair matters, so these are not discarded.

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

$$d_m(S_i, T_j) = \begin{cases} \dfrac{\max\{|\mathcal{S}|, |\mathcal{T}|\} - |r_{\varnothing left}(S_i) \cap r_{\varnothing left}(T_j)| - |r_{right}(S_i) \cap r_{right}(T_j)|}{\max\{|\mathcal{S}|, |\mathcal{T}|\}}, & \text{if} E_{S_i} = E_{T_j} \\ 1, & \text{if} E_{S_i} \neq E_{T_j} \end{cases}$$

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 $\mathcal{T}'$. Problem 3.1 can be formalized in the present context as the following optimization problem:

Problem 4.1. ***(The Assignment Problem)*** *Given $\mathcal{S}'$, $\mathcal{T}'$, and $D_{\mathcal{S}',\mathcal{T}'}$, assign each event-interval in $\mathcal{S}'$ to exactly one event-interval in $\mathcal{T}'$ so that the total assignment cost is minimized.*

Problem 4.1 can be solved by the Hungarian algorithm. 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. The assignment cost $C(\mathcal{S}', \mathcal{T}')$ corresponds to the distance, called *Artemis Distance*, between the original e-sequences $\mathcal{S}$ and $\mathcal{T}$.

Definition 4.6. ***(Artemis Distance)*** *Given $\mathcal{S}'$, $\mathcal{T}'$, and $\mathcal{H}(\mathcal{S}', \mathcal{T}')$, the* Artemis distance *of $\mathcal{S}$ and $\mathcal{T}$ is defined as follows:*

$$Artemis(\mathcal{S}, \mathcal{T}) = \sum_{i=1}^{|\mathcal{S}'|} d_m(S'_i, h(S'_i)). \tag{4.10}$$

The whole procedure for calculating the Artemis Distance of two event-interval sequences is depicted using pseudocode in Algorithm 2.

Example. Figure 4.5 shows two e-sequences $\mathcal{S}$ and $\mathcal{T}$. The event-interval relation sets will be first computed by the *mapping step*. For $\mathcal{S}$, these sets are: $\{follow(\varnothing, A), overlap(A, B), follow(A, C)\}$ for A, $\{follow(\varnothing, B), overlap(A, B), overlap(B, C)\}$ for B, and $\{follow(\varnothing, C), follow(A, C), overlap(B, C)\}$ for C. For $\mathcal{T}$, these sets are $\{follow(\varnothing, A), follow(A, B), follow(A, D)\}$ for A, $\{follow(\varnothing, B), follow(A, B), overlap(B, D)\}$ for B, and $\{follow(\varnothing, D),$

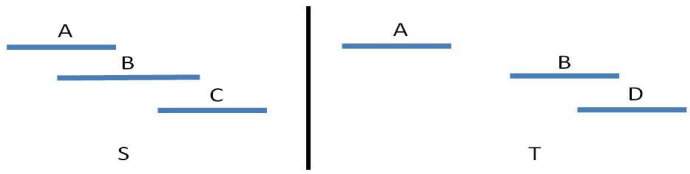


Figure 4.5: Two e-sequences $\mathcal{S}$ and $\mathcal{T}$ used as an example for `Artemis`.

$follow(A, D)$, $overlap(B, D)\}$ for D. At the *matching step* the Hungarian algorithm would return $\mathcal{H}(\mathcal{S}, \mathcal{T}) = (A, B, D)$. Finally, $Artemis(\mathcal{S}, \mathcal{T}) = (2/3 + 2/3 + 1) = 7/3$.

If the distance of event-intervals ($d_m(S_i, T_j)$) were calculated simply by the following equation:

$$d_m(S_i, T_j) = \begin{cases} \dfrac{\max\{|\mathcal{S}|, |\mathcal{T}|\} - |r(S_i) \cap r(T_j)|}{\max\{|\mathcal{S}|, |\mathcal{T}|\}}, & \text{if} E_{S_i} = E_{T_j} \\ 1, & \text{if} E_{S_i} \neq E_{T_j} \end{cases} \tag{4.11}$$

then `Artemis` would be unable to distinguish between the event-interval sequences in Figure 4.6, since their distance would be zero.

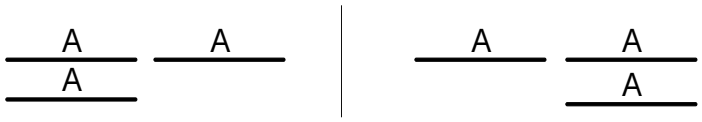


Figure 4.6: A pair of e-sequences that would cause `Artemis` to yield zero distance, if the distance of event-intervals were computed using Equation 4.11.

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 $O(m^3)$ [11]. A lower bound for speeding up the computation of `Artemis` is described next.

Claim 4.1. *`Artemis` does not violate the identity of indiscernibles.*

Proof. The proof is trivial. In order to have $Artemis(\mathcal{S}, \mathcal{T}) = 0$, the two e-sequences must have the same count of each label. Otherwise, the Hungarian algorithm would match at least one pair of event-intervals with a score of 1. In addition, the interval labels must appear in the same order. Otherwise, in the matching, the sets $r_{\varnothing left}$ and r_{right} would not have equal cardinality for at least one of the matched pairs, which would not allow a zero overall score. Finally, for the same reason, for each pair of matched event-intervals, the sets $r_{\varnothing left}$ and r_{right} must contain the same relations. As a result, the event-intervals of the two e-sequences must appear in the same order and event-intervals in the same position must have the same relations with any of the other event-intervals in the same position. This implies that $Artemis(\mathcal{S}, \mathcal{T}) = 0$ if and only if $\mathcal{S}$ and $\mathcal{T}$ are identical. □

4.4 Lower Bounding `Artemis`

The proposed 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.

Algorithm 2 Computing `Artemis`

Input: Two Event-Interval Sequences $\mathcal{S}, \mathcal{T}$
Output: The distance score: $Artemis(\mathcal{S}, \mathcal{T})$

```
/* Compute relation sets for all event-intervals in S */
for all intervals S_i ∈ S do
    r_∅(S_i) = R_l(∅, S_i)
    r_left(S_i) = ∅
    for all intervals S_j ∈ S s.t. j < i do
        r_left(S_i) = r_left(S_i) ∪ R_l(S_j, S_i)
    end for
    r_right(S_i) = ∅
    for all intervals S_j ∈ S s.t. j > i do
        r_right(S_i) = r_right(S_i) ∪ R_l(S_i, S_j)
    end for
end for
/* Compute relation sets for all event-intervals in T */
for all intervals T_i ∈ T do
    r_∅(T_i) = R_l(∅, T_i)
    r_left(T_i) = ∅
    for all intervals T_j ∈ T s.t. j < i do
        r_left(T_i) = r_left(T_i) ∪ R_l(T_j, T_i)
    end for
    r_right(T_i) = ∅
    for all intervals T_j ∈ T s.t. j > i do
        r_right(T_i) = r_right(T_i) ∪ R_(T_i, T_j)
    end for
end for
S', T' = Augment(S,T)
/* Compute all event-interval distances d_m(S'_i, T'_j) */
for all S'_i ∈ S' do
    for all T'_j ∈ T' do
        DistanceMatrix(i, j) = d_m(S'_i, T'_j)
    end for
end for
H(S', T') = Min_Cost_Maximum_Bipartite_Matching(DistanceMatrix)
return ∑_{i=1}^{|S'|} d_m(S'_i, h(S'_i))
```

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 4.1. *Given $\mathcal{S}$ and $\mathcal{T}$, a lower bound for $Artemis(\mathcal{S},\mathcal{T})$ is given by*

$$Artemis_{LB}(\mathcal{S},\mathcal{T}) = \frac{k}{2} + \left(m - \frac{k}{2}\right)\left(\frac{k}{2m}\right) = k - \frac{k^2}{4m}, \tag{4.12}$$

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

Proof. Knowing that $||v^{\mathcal{S}} - v^{\mathcal{T}}||_1 = k$ we can be sure that $Artemis(\mathcal{S},\mathcal{T}) \geq k/2$; those k event-intervals, $k/2$ in each e-sequence, are matched with each other contributing a score of $d_m(S_i, T_j) = 1$ for each of the $k/2$ pairs. If $k/2$ is equal to the length of the e-sequences, then $Artemis(\mathcal{S},\mathcal{T}) = k/2$. However, if the opposite case holds, 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. So, given that $m = max(|\mathcal{S}|, |\mathcal{T}|)$ and $m > \frac{k}{2}$, 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 event-intervals. Thus, yielding adding to $Artemis(\mathcal{S},\mathcal{T})$ 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 three datasets in Section 6.2.

Chapter 5

Longest Common Sub-Pattern

In this section we study the problem of finding the Longest Common Sub-pattern (LCSP) between a pair of arrangements of temporal intervals. We formally define the notion of the LCSP and prove that finding it belongs to the class of NP-complete problems. Furthermore, we present an exact algorithm to retrieve the LCSP of pairs of arrangements, which is based on dynamic programming.

Definition 5.1. *Given two temporal interval arrangements, $\mathcal{A}$ and $\mathcal{B}$, their Longest Common Sub-pattern is the maximum ordered set of intervals $\mathcal{S}_{LCSP} = \{ S_{LCSP1}, S_{LCSP2}, \cdots, S_{LCSPk}\}$, such that there exist event-intervals $\{S_{a1}, S_{a2}, \cdots, S_{ak}\}$ in $\mathcal{A}$, and $\{S_{b1}, S_{b2}, \cdots, S_{bk}\}$ in $\mathcal{B}$, and it holds: $E_{SLCSPi} = E_{Sai} = E_{Sbi}$, $1 \leq i \leq k$ and $R(E_{SLCSPi}, E_{SLCSPj}) = R(E_{Sai}, E_{Saj}) = R(E_{Sbi}, E_{Sbj}), \forall i, j$ s.t. $1 \leq i < j \leq k$.*

In other words, the LCSP of a pair of arrangements, $\mathcal{A}$ and $\mathcal{B}$, is an arrangement with its intervals being a subset of those present in both $\mathcal{A}$ and $\mathcal{B}$. Furthermore, for any pair of intervals in the LCSP, the corresponding pairs of intervals in $\mathcal{A}$ and $\mathcal{B}$ have the same type of relation. For example, in Figure 5.1, the LCSP of the two arrangements is the pattern formed by intervals A, C, D.

Figure 5.1: The Longest Common Sub-pattern of the two temporal interval arrangements is the sub-arrangement formed by the intervals with labels A,C and D.

Theorem 5.1. *The problem of finding the LCSP of two temporal interval arrangements is NP-complete.*

Proof. We prove the theorem by first proving the following lemma.

Lemma 5.1. *The Clique decision problem can be reduced to the LCSP problem.*

If we prove the lemma, we have proved that LCSP is at least as hard as the Clique decision problem, which is NP-complete [20]. Thus, LCSP would belong to the NP-hard class of problems. First, we demonstrate how any undirected graph $G = (V, E)$ can be encoded as an e-sequence. After that, the e-sequence can be trivially transformed into an arrangement simply by disregarding the absolute values of the time points (but maintaining their order).

In the transformation of graphs into e-sequences, each vertex $u_i \in V$ corresponds to a time point t_i; the correspondence between vertices and time points is not important as long as it is consistent throughout the procedure. For every edge $e = (u_i, u_j) \in E$ we create interval $S = (a, t_i, t_j)$ (if $t_i < t_j$, else $S = (a, t_j, t_i)$). There is no reason to specifically select 'a' as a label, but it is important that all labels are the same. Unconnected vertices do not affect the solution of Clique so they can be discarded; alternatively, they can be conserved by creating instantaneous intervals of the form $(a.t_i, t_i)$. Note that the reverse procedure, from e-sequences to graphs, would create undirected multigraphs with labeled edges.

Clique can be reduced to LCSP as follows. The graph G is transformed into an arrangement $\mathcal{A}_G$ as described above. Given the parameter k we create a second sequence $\mathcal{A}_k$ that corresponds to a clique of size k. This is easily achieved by creating all possible $k(k-1)/2$ intervals of the form $S = (a, t_i, t_j)$, with $1 \leq i \leq k-1$, $i < j \leq k$. The result of LCSP determines the result for Clique. The whole arrangement $\mathcal{A}_k$ is found in $\mathcal{A}_G$, or equivalently the size of the LCSP between $\mathcal{A}_G$ and $\mathcal{A}_k$ is equal to $k(k-1)/2$, if and only if the graph contains a clique of size k. Proving the last statement:

($\Leftarrow$) If graph G has a clique of size k, then the size of the LCSP is $k(k-1)/2$: If graph G has a clique of size k, then there exist k vertices that are fully connected. Suppose the vertices of the clique are $V_{clique} = \{u_{c1}, ...u_{ck}\}$. The reduction would create $\mathcal{A}_G$ containing, among others, all $k(k-1)/2$ intervals of the form $(a, t_{u_{c}i}, t_{u_{c}j})$, with $1 \leq i \leq k-1$, $i < j \leq k$. The reduction would also create $\mathcal{A}_k$, which contains exactly $k(k-1)/2$ intervals in a pattern identical to that formed by the intervals corresponding to the clique. Thus, the size of the LCSP would be equal to $k(k-1)/2$.

($\Rightarrow$) If the size of the LCSP is $k(k-1)/2$, then the graph G has a clique of size k: If the size of LCSP is equal to $k(k-1)/2$ then there exists a set of intervals in $\mathcal{A}_G$ which create a pattern identical to that of $\mathcal{A}_k$. That means there exist $k(k-1)/2$ intervals in $\mathcal{A}$ of the form $S = (a, t_i, t_j)$ for all $1 \leq i \leq k-1$, $i < j \leq k$. Thus, k vertices exist in G connected by edges (u_i, u_j) for all $1 \leq i \leq k-1$, $i < j \leq k$. Equivalently, k nodes in G are fully connected.

Proposition 5.1. *The decision version of LCSP belongs to the NP complexity class.*

It is trivial to show that a candidate solution can be verified in polynomial time, based on the Definition 5.1.

We have shown that LCSP $\in NP$ and that every problem in NP reduces to LCSP. As a consequence, LCSP is an NP-complete problem.

□

5.1 An exact algorithm for LCSP

We present an exact algorithm, based on dynamic programming, to retrieve the LCSP between pairs of temporal interval arrangements. Given $\mathcal{A} = \{S_{A1}, \cdots, S_{Am}\}$ and $\mathcal{B} = \{S_{B1}, \cdots, S_{Bn}\}$, we denote by $LCS(i,j), 1 \leq i \leq |\mathcal{A}|, 1 \leq j \leq |\mathcal{B}|$, the LCSP between $\{S_{A1}, \cdots, S_{Ai}\}$ and $\{S_{B1}, \cdots, S_{Bj}\}$ where the sub-pattern must include an interval corresponding to S_{Ai} and S_{Bj}; if $E_{SAi} \neq E_{SBj}$ then $LCS(i,j) = \emptyset$. The idea behind this algorithm is that we can infer $LCS(i,j)$ if we know all previous $LCS(p,q)$, with $1 \leq p \leq i$ and $1 \leq q \leq j$.

Unlike the case of LCS for strings, in temporal interval arrangements it is not sufficient to know only $LCS(i-1, j-1)$, $LCS(i-1, j)$ and $LCS(i, j-1)$. Furthermore, given $LCS(p,q)$, with $p < i$ and $q < j$, it is not sufficient to simply append the new interval. Actually, it is not even correct. The reason is that all other previous intervals do not necessarily have a *follow* relation with S_{Ai} and S_{Bj}. For example, in Figure 5.1 the interval labeled "D" has a different relation with the "B" interval in the two arrangements respectively, although "C" is the previous of "D". Thus, it is not sufficient to check just the relation with the last interval. Instead, S_{Ai} and S_{Bj} must be checked against all intervals of the sub-solution and keep only

those that have the same relation (thus, it is possible to have $|LCS(p,q)| \geq |LCS(i,j)|$). So, for each $L(i,j)$ all $L(p,q)$ must be examined to discover the one that yields the actual longest CSP. Additionally, for a single $LCS(i,j)$, multiple solutions may exist, e.g. for $L(3,3)$ the two solutions are $\{A,C\}$ and $\{B,C\}$. None of the solutions can be discarded since it is not clear which one would be the appropriate choice for the next sub-problems.

In a general form, the described approach can be expressed as:

$$LCS(i,j) = \begin{cases} \emptyset, & \text{if } E_{SAi} \neq E_{SBj} \\ \max\limits_{\substack{p \leq i, q \leq j \\ (p,q) \neq (i,j)}} \{LCS(p,q) \otimes (i,j)\}, & \text{else} \end{cases}$$

where $LCS(p,q) \otimes (i,j)$ is the arrangement that occurs from the interval corresponding to S_{Ai} and S_{Bj} and the event-intervals in $LCSP(p,q)$ that have the same relations with intervals S_{Ai} and S_{Bj} in their respective arrangements. The LCSP of two arrangements can be computed as described in Algorithm 3.

Algorithm 3 Longest Common Sub-pattern of Temporal Interval Arrangements

```
for i = 1 to |A| do
  for j = 1 to |B| do
    if label(S_Ai) ≠ label(S_Bj) then
      LCS(i,j) = ∅
    else
      for all sub-problems (p,q) of (i,j) do
        for all solutions S_k(p,q) of LCS(p,q) do
          newLCS = S_k(p,q) ⊗ (i,j)
          if |newLCS| > |S_m|, S_m ∈ LCS(i,j) then
            LCS(i,j) = {newLCS}
          else if |newLCS| = |S_m| then
            LCS(i,j) = LCS(i,j) ∪ {newLCS}
          end if
        end for
      end for
    end if
  end for
end for
```

Claim 5.1. *Algorithm 3 returns the LCSP of a pair of arrangements of temporal intervals.*

Proof. First we must prove that $L(i,j)$ returns the LCSP for the corresponding sub-problem .

For any $1 \leq i \leq m, 1 \leq j \leq n$:

- If $E_{SAi} \neq E_{SBj}$ then $LCS(i,j) = \emptyset$. If the two intervals do not have the same label, then it is not possible to have an LCS with its last interval corresponding to S_{Ai} and S_{Bj}, since no labeled interval can be identical to two intervals of different labels.
- If $E_{SAi} = E_{SBj}$ then $LCS(p,q) \otimes (i,j)$ yields a common sub-pattern of $\{E_{SA1}, ..., E_{SAi}\}$ and $\{E_{SB1}, ..., E_{SBj}\}$: By selecting only the intervals that induce similar relations, we make sure that the correspondent to i and j has the same relations to the previous intervals. Conversely, the existing intervals have the same relations to the correspondent of i and j. Additionally, existing intervals of LCS have identical relations with

their correspondents in $\mathcal{A}$ and $\mathcal{B}$; this was examined when each interval was added to the solution of the previous sub-problems.

- If $E_{SAi} = E_{SBj}$ and all previous sub-problems $LCS(p, q)$ yield $\emptyset$ as their solution, then $LCS(i, j)$ is composed only of an interval corresponding to S_{Ai} and S_{Bj}: Supposing that $LCS(i, j)$ was composed of more than one interval, then there must exist a pair of intervals with the same label in $\{S_{A1}, .., S_{Ai-1}\}$ and $\{S_{B1}, .., S_{Bi-1}\}$. That is a contradiction since it would imply that not all previous sub-problems yield $\emptyset$ as their solution.

- If $E_{SAi} = E_{SBj}$ then the $\otimes$ operation yields a LCSP: Suppose that the LCSP is correctly retrieved for all previous sub-problems $LCS(p, q)$, but not for $LCS(i, j)$. This would imply that an interval belonging to the LCSP of $\{E_{SA1}, ..., E_{SAi}\}$ and $\{E_{SB1}, ..., E_{SBj}\}$ exists but was not selected. If the not-selected interval belongs to the LCS of $\{E_{SA1}, ..., E_{SAi}\}$ and $\{E_{SB1}, ..., E_{SBj}\}$ then it has the same relation to S_{Ai} and S_{Bj}. But then, since the relations are the same, the interval would have been selected for $LCS(i, j)$, which contradicts to the previous. Thus, the algorithm at point (i, j) returns the LCSP of $\{E_{SA1}, ..., E_{SAi}\}$ and $\{E_{SB1}, ..., E_{SBj}\}$.

Since the method at point (i, j) returns the LCSP for the corresponding sub-problem, then the LCSP of the two whole arrangements is found by selecting $LCS(i, j)$ which yields the largest value. □

Complexity In the above approach, $LCS(i, j)$ must be computed for all possible pairs of i and j. To compute each $LCS(i, j)$, it is necessary to check all the solutions of all $O(|A|\cdot|B|)$ sub-problems; checking a solution requires linear time with respect to the size of the LCS, which is at most $\min\{|A|, |B|\}$. So, the total complexity of the algorithm is $O(n^3 \cdot m^2 \cdot s)$, if $n \leq m$, where s is the maximum number of solutions over all $LCS(i, j)$. This does not prove that we have found a polynomial time algorithm for NP-complete problems, since the number of solutions s can be exponential in the size of the input.

Computationally hard instances of Clique reduce to hard instances of LCSP. In such cases, the exact algorithm has to search among a number of solutions which is exponential in the size of the arrangements. This becomes evident as for every sub-problem, the labels of the pair of the corresponding intervals are the same and all previous solutions must be examined. Despite that, this is not a characteristic only of arrangements containing intervals which all have the same label.

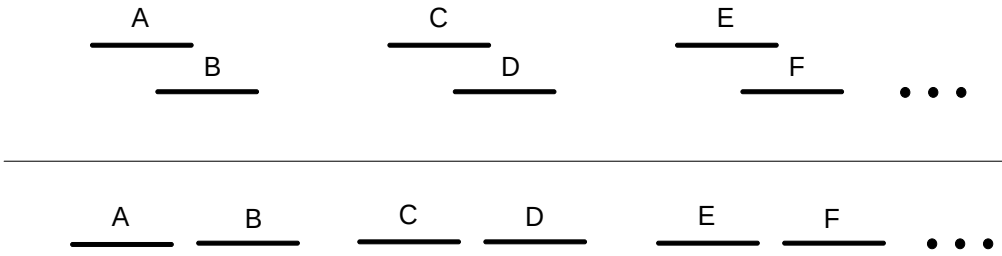


Figure 5.2: A category of arrangement pairs where any event-interval label appears exactly once, yet the number of candidate solutions stored at each point of the dynamic programming algorithm is exponential in the size of the arrangements.

In Figure 5.2 we display a category of instances where any event-interval label appears exactly once, yet the number of candidate solutions stored at each point of the dynamic programming algorithm is exponential in the size of the arrangements. In particular, the number of partial solutions is $O(2^{\frac{n}{2}})$. There exist multiple LCSPs with size equal to half the size of the original arrangements. They contain only one event-interval for every pair of overlapping event-intervals in the first arrangement.

Chapter 6

Experiments

In this chapter we attempt to obtain an insight into the behavior of the three proposed distance functions and the lower bounding technique for `Artemis`, which were presented in Chapter 4. In Section 6.1 we describe our methodology and the performed experiments, while the results are demonstrated in Section 6.2. Finally, our conclusions are presented in Section 6.3.

6.1 Experimental Setup

In Kostakis et al. [26] we have demonstrated, through thorough experimentation, the superiority of the normalized Manhattan distance over its simpler form and the Frobenius norm. Due to that, experiments in this thesis are limited only to the use of the normalized version of the distance measure. We refer to the measure using the term "Relation Matrix" and by that we mean the whole procedure with the use of the normalized version when comparing the matrices.

The proposed distance functions have been benchmarked on three real datasets (Details regarding the datasets are displayed in Table 6.1):

- **Hepatitis** [42] The dataset contains information about 498 patients who have either Hepatitis B or Hepatitis C. The event-intervals represent the results of 25 regular tests.
- **ASL** [41] The event labels in this dataset are transcriptions from videos of American Sign Language expressions provided by Boston University.
- **Pioneer** [35] This dataset was constructed from the Pioneer-1 dataset available in the UCI repository. Each e-sequence in the dataset describes one of three scenarios: *gripper*, *move*, *turn*. The event-intervals correspond to the input provided by the robot sensors.

The distance functions were evaluated with respect to robustness against two types of artificial noise, k-NN classification accuracy, and scalability. In addition, the efficiency of $Artemis_{LB}$ was tested by computing its *tightness* and *pruning power* on 1-NN queries. These two metrics are commonly used in the evaluation of lower bounds (e.g., Keogh et al. [21]).

Noise robustness. The robustness in the presence of noise was tested by using two types of artificial noise. In the first type of noise, each event-interval within an e-sequence is independently shifted by an offset back or forth in time. Whether an event-interval will be shifted is determined by the *offset probability* value. In addition, the limits on the value of the offset are given by the *distortion level d*. The distortion level is given as a percentage of the length of the whole e-sequence. For each event-interval, a random value under the uniform distribution is chosen in that integer interval to determine the offset. An event-interval has equal probability to be shifted either back or forth in time. The durations of

Dataset	E-Sequences	Total Intervals	Min	Max	Average	Labels ($\|\sigma\|$)	Classes
ASL	873	15675	4	41	17.95	216	5
Hepatitis	498	53921	15	592	108.27	147	2
Pioneer	160	8949	36	89	55.93	92	3

Table 6.1: Dataset Summary.

the event-intervals remained unaffected. This type of artificial noise attempts to simulate noisy sources or recording devices. Real-world cases for this could include humans who learn American sign language or who rehabilitate from brain injuries, or noise in sensor networks. The distortion level and offset probability parameter values were set to 0.2 to 1, with step 0.2.

A drawback of the first noise type is that the shifting of event-intervals could result to semantic invalidity, e.g. "robot walks forward" overlaps with "robot walks backwards". To avoid such cases, we further experiment with artificial noise which is based on swaps of event-interval labels, while the durations and the relations of the event-intervals remain unaffected. Given an e-sequence, the *swap probability* parameter determines if an event-interval will have its label swapped with that of another event-interval. The other event-interval is chosen uniformly at random from the whole e-sequence. The swap probability parameters values tested were 0.2 to 1, with step 0.2. We must note that, in all cases, the noise is added off-line and that all methods were tested on exactly the same distorted e-sequences. This happened in order to rule out differences in score caused by the randomness factor.

Noise is incorporated into e-sequences which then serve as the queries in 1-NN search tasks. Given a database of event-interval 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. We compared the three distance functions in terms of: *retrieval accuracy* (the fraction of noisy queries for which the originating e-sequence is retrieved) and *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).

Classification. For the k-NN classification experiments, we studied the distance functions' efficiency under 1-NN and 3-NN classifiers. Our datasets contain labeled samples; the samples belong to predefined classes and the class of each sample is known (The number of classes for each dataset is depicted in Table 6.1). In this experiment, we classify each sample based on the class labels of its nearest neighbors. The neighbors are determined by the the distance function. In the 1-NN case the sample is assigned to the class of its nearest neighbor, while in the case of 3-NN the sample is assigned to the class which is the majority element of the classes of the 3 nearest neighbors. A classification of a sample is considered correct if the class assignment agrees with the true class of the sample. The *accuracy* of each classifier is determined as the ratio of correct class assignments.

Due to the fact that the e-sequences in the ASL dataset can belong to up to 5 classes (wh-question, etc) simultaneously, while some e-sequences do not belong to any class (simple affirmative phrases), we had to modify the accuracy evaluation metric. Only for the ASL dataset, we considered two cases, one that classifies the whole dataset and another that focuses on the classification of samples that have non-empty label sets. In both cases, for each sample the classifier returns a score in $[0, 1]$ denoting the average ratio of common class-labels with its k nearest neighbors. For example, a phrase with labels $\{a\}$ and neighbors with label sets $\{a\}$, $\{a, b\}$ and $\emptyset$ respectively, would yield $(1 + 0.5 + 0)/3$. Thus, for the ASL dataset the accuracy of the classifier is the average score over all samples.

We designed an additional experiment specifically for the ASL dataset. There exist in total 288 e-sequences in the dataset which portray exactly the same phrase in English with

one or more other e-sequences. We study what ratio of the 288 equivalent e-sequences are retrieved by examining the k nearest neighbors of every sample, for all possible values of k. We monitor two values. The first is the ratio of the 288 e-sequences for which an identical phrase can be detected within their k nearest neighbors. The second is the total sum of identical phrases that can be detected within the k nearest neighbors. The two values would be the same if any phrase in English were to be represented by at most two e-sequences in the dataset. This experiment was designed to investigate the suitability of the methods in the field of language technology and relevant real-world applications.

Scalability. In the attempt to evaluate the scalability of the proposed distance functions, we embedded time-monitoring functions in our implementations. For each e-sequence, we count the time needed to transform the e-sequence in the right form (vector form for DTW, enumerate the relation sets for `Artemis`, creation of relation matrix) and then compare it against the whole dataset. The dataset, serving as the database, had already been transformed in the appropriate forms in advance. For the case of `Artemis`, we have not implemented the use of hashing, thus the complexity is $O(n^4)$ with respect to the size of the e-sequences. Our experiments were implemented in Java and were performed on a PC running Ubuntu Linux, equipped with Intel Core 2 Duo 2GHz CPU and 4GB RAM.

Lower bounding. To assess the quality of $Artemis_{LB}$, we compute its pruning power for 1-NN queries in the database (see Algorithm 4) and its tightness. The pruning power is defined as the ratio of pruned comparisons, using the `Artemis`, over the total number of comparisons that would have been required if the database were to be serially scanned. The tightness is defined as the average ratio of the lower bound distance over the distance given by `Artemis`.

Algorithm 4 1-NN Search Using $Artemis_{LB}$

$d_{NN} = \infty$
for each S_i in DB **do**
  //Check Lower Bound distance
  if $Artemis_{LB}(S_i,Q) < d_{NN}$ **then**
    //Check distance given by `Artemis`
    if $Artemis(S_i,Q) < d_{NN}$ **then**
      $d_{NN} = Artemis(S_i, Q)$
    end if
  end if
end for

6.2 Results

Figures 6.1, 6.2 and 6.3 depict the results of the noise robustness experiment for the three datasets, when the noise appears as shifts of the event-intervals. When tested on the Pioneer dataset (results in Figure 6.3) `Artemis` achieved 100% retrieval accuracy for all pairs of noise probability and distortion level values. Due to that, we omit the subfigure of the comparison. The results of the noise robustness experiments for the second type of artificial noise are depicted in Figures 6.4, 6.5 and 6.6.

The results of the tests on the three distance functions for their robustness against noise depict `Artemis` to be a clear winner. The Relation Matrix approach performed significantly better than DTW, achieving high retrieval accuracy rates but did not consistently achieve the success rates of `Artemis`. The DTW vector-based approach displayed a significant decline in performance with the increase in the value of the offset or swap probability and distortion level parameters. `Artemis` was able to maintain a very high rate of finding the noisy queries' originating counterparts. For all but one of the datasets and experiments its success rate was over 98%. `Artemis` displayed a decline in performance only for the Pioneer dataset and the noise type that consists of swapping the labels (Figure 6.6). Similarily, the rate for Relation Matrix was over 85% for all the cases excluding the Pioneer dataset with swaps-based noise. To summarize, in all the noise robustness results, `Artemis`' performance was either identical to or better than the performance of Relation Matrix and DTW.

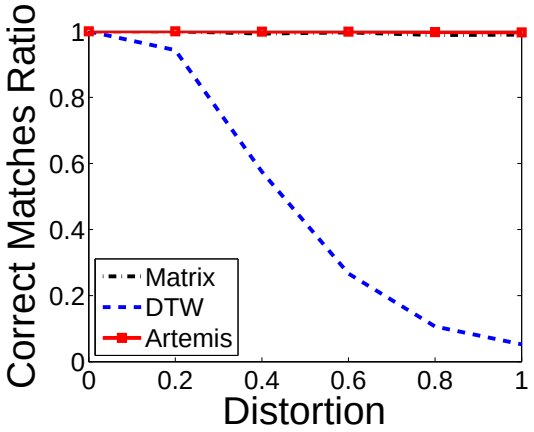
The results of the k-NN classification are depicted on Table 6.2. The values in the cells denote the classification accuracy of each classifier on each dataset. For the Hepatitis dataset, the 3-NN DTW-based classifier performs marginally better than the rest. `Artemis` and Relation Matrix perform better than DTW for the Pioneer dataset, but contrary to the Relation Matrix, `Artemis` maintains its high classification accuracy in the 3-NN setting. For the ASL dataset, the Relation Matrix approach is the best, while DTW ranks second and `Artemis` third. However, for the pruned ASL dataset (only samples of non-empty label-sets) the situation is reversed. Additionally, we examined the clustering performance of LCSP by using our exact algorithm. The only purpose was to study whether the LCSP was a more meaningful measure for clustering any of the datasets, in comparison to the pairwise distance. The Hepatitis dataset contained hard instances which resulted in almost exponential memory requirements (over 4GB for the comparison of two arrangements with 250 intervals each). Due to that, we were not able to derive any results for that dataset.

Dataset	`Artemis` 1-NN	`Artemis` 3-NN	DTW 1-NN	DTW 3-NN	Matrix 1-NN	Matrix 3-NN	LCSP 1-NN	LCSP 3-NN
HepData	0.7209	0.7811	0.7403	0.8072	0.7490	0.7831	-	-
Pioneer	0.9750	0.9750	0.93750	0.9375	0.9750	0.9688	0.9750	0.9625
ASL	0.4307	0.4013	0.4358	0.4188	0.4404	0.4230	0.3418	0.3248
ASL(pruned)	0.3464	0.3402	0.3020	0.2893	0.2813	0.2733	0.2400	0.2265

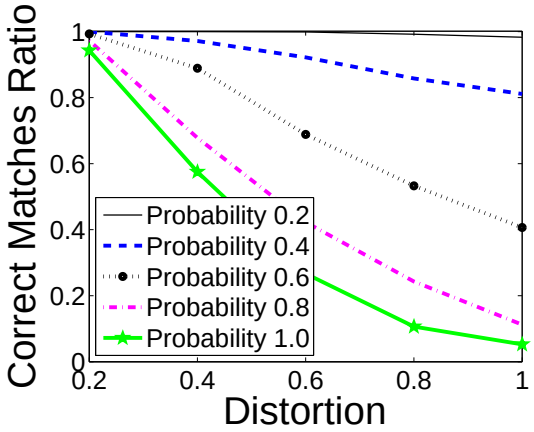
Table 6.2: k-NN classification results.

Figure 6.7 depicts the results of the experiment, conducted on the ASL dataset, in which the k nearest neighbors of each e-sequence are scanned to determine if they denote the same phrase in English. Figure 6.7a shows the ratio (of the e-sequences which do have equivalents) for which at least one identical phrase can be found within k nearest neighbors. The ratio of the total sum of existing identical phrases that have been found for each method is displayed in Figure 6.7b. For both cases, Relation Matrix returns a better grouping of semantically related e-sequences. Additionally, `Artemis` performs better than DTW.

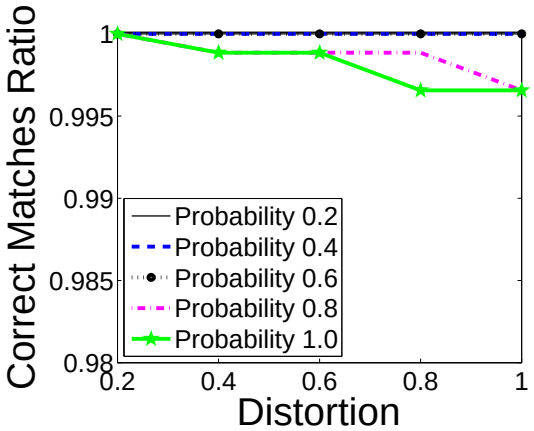
The results of the scalability results can be seen in Figures 6.8, 6.9 and 6.10. In Figure 6.8 we observe that `Artemis` is slower than the Relation Matrix and vector-based DTW approaches, as the complexity analysis of Chapter 4 would suggest. On the other hand, we observe, in Figure 6.9, that for the case of ASL, `Artemis` is faster than the Relation Matrix and DTW measures. The reason for that is the large size of the alphabet ($|\sigma|$). In Figure 6.10 we witness a blow up in the run times of `Artemis`. For e-sequences of large size, `Artemis`' $O(n^4)$ complexity (without the use of hashing) significantly affects its performance.



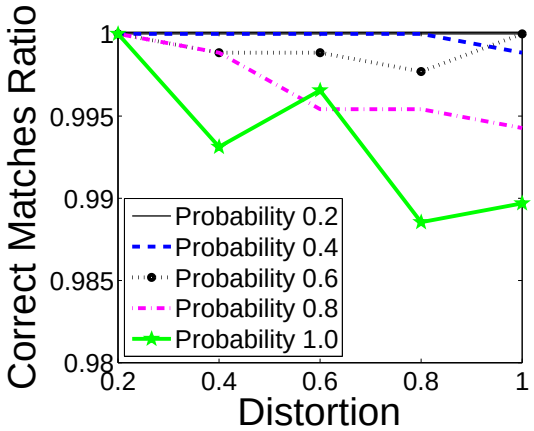
(a) Retrieval Accuracy. Offset probability 1.0



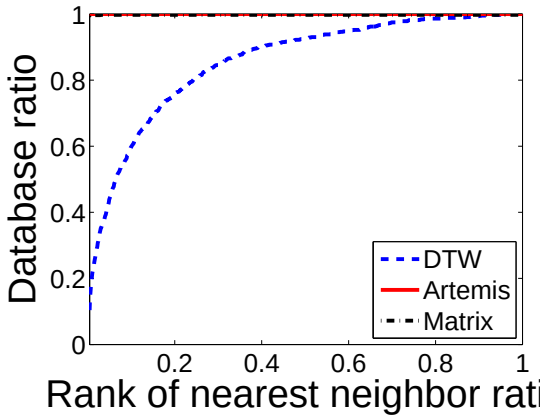
(b) DTW Retrieval Accuracy comparison



(c) `Artemis` Retrieval Accuracy comparison

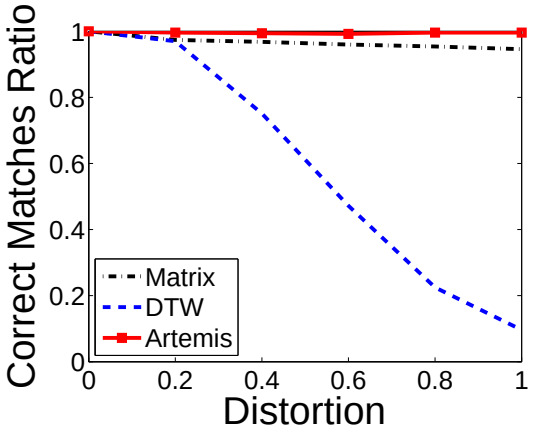


(d) Relation Matrix Retrieval Accuracy comparison

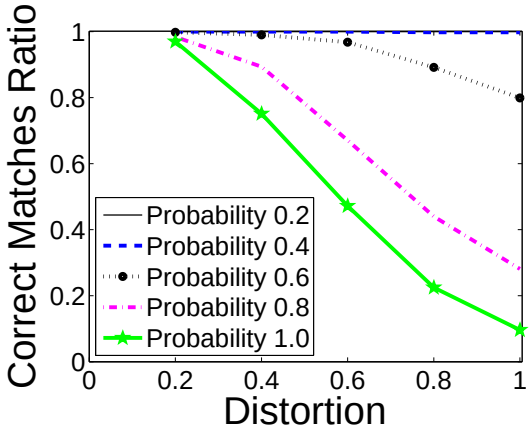


(e) Ranks of NN. Offset probability 1.0, distortion 1.0

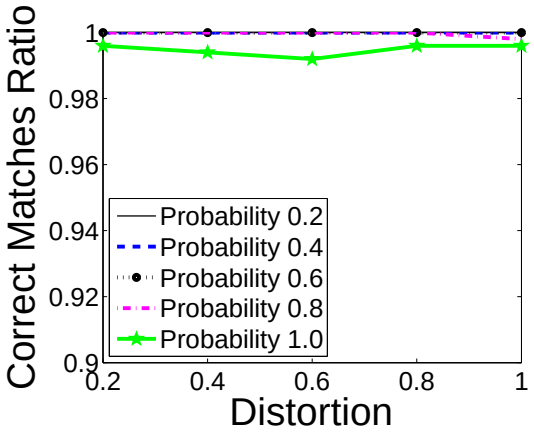
Figure 6.1: Offset noise robustness experiment. ASL dataset.



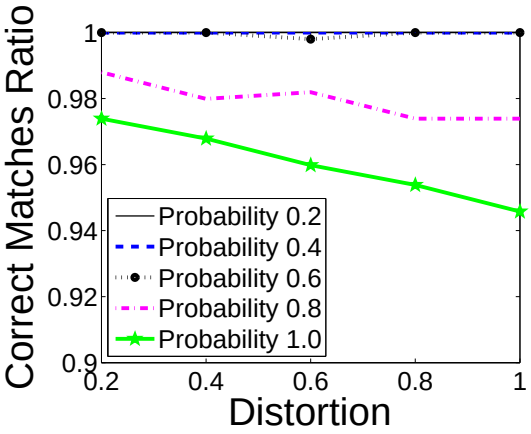
(a) Retrieval Accuracy. Offset probability 1.0



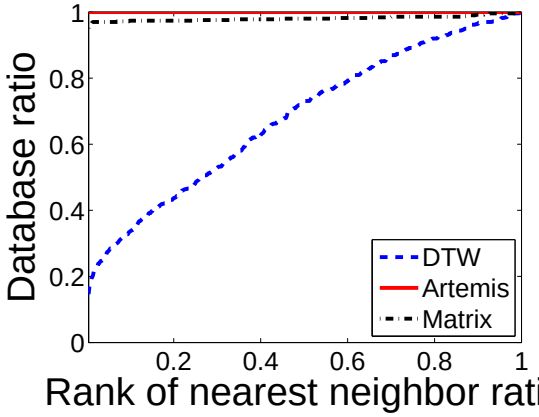
(b) DTW Retrieval Accuracy comparison



(c) `Artemis` Retrieval Accuracy comparison

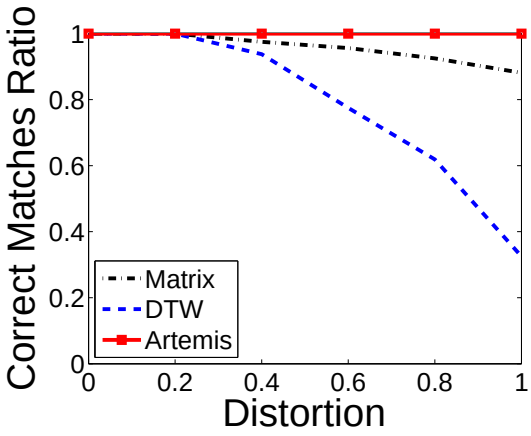


(d) Relation Matrix Retrieval Accuracy comparison

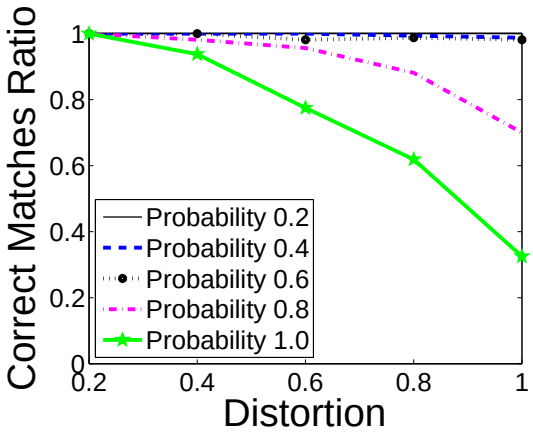


(e) Ranks of NN. Offset probability 1.0, distortion 1.0

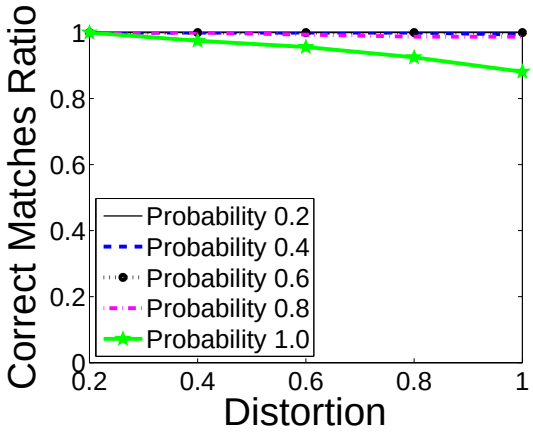
Figure 6.2: Offset noise robustness experiment. Hepatitis dataset.



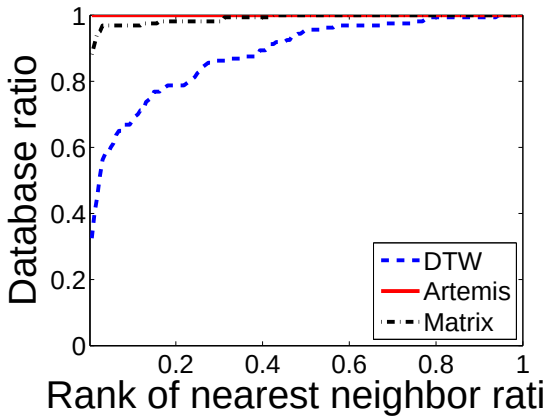
(a) Retrieval Accuracy. Offset probability 1.0



(b) DTW Retrieval Accuracy

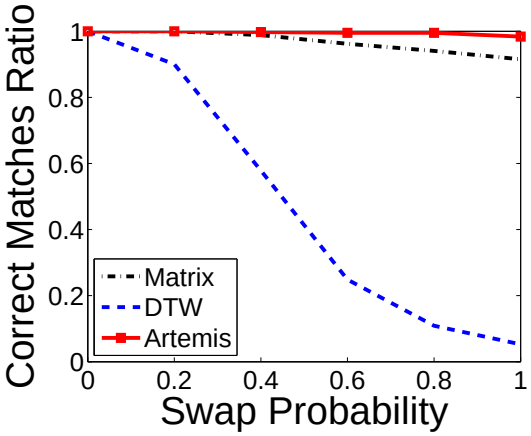


(c) Relation Matrix Retrieval Accuracy

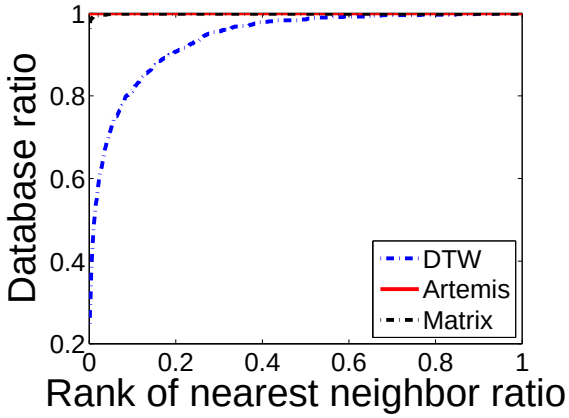


(d) Ranks of NN. Offset probability 1.0, distortion 1.0

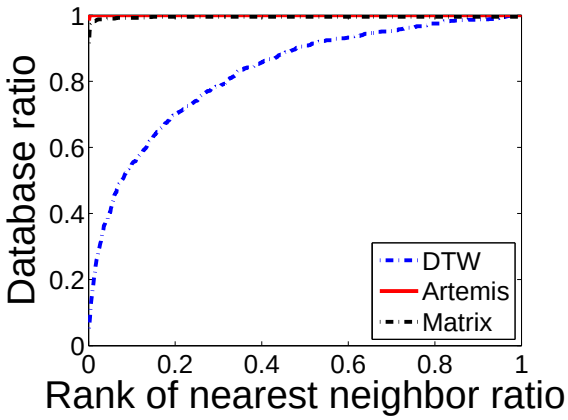
Figure 6.3: Offset noise robustness experiment. Pioneer dataset.



(a) Retrieval Accuracy.

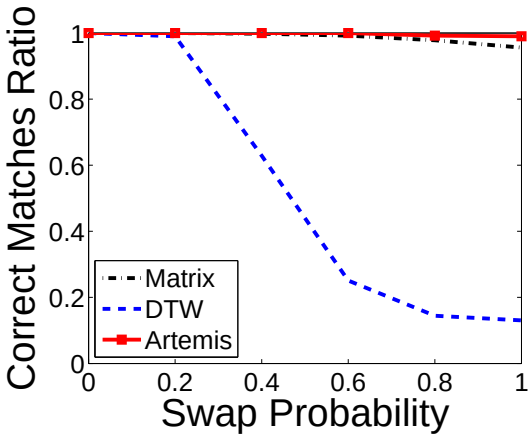


(b) Ranks of NN. Swap probability 0.6

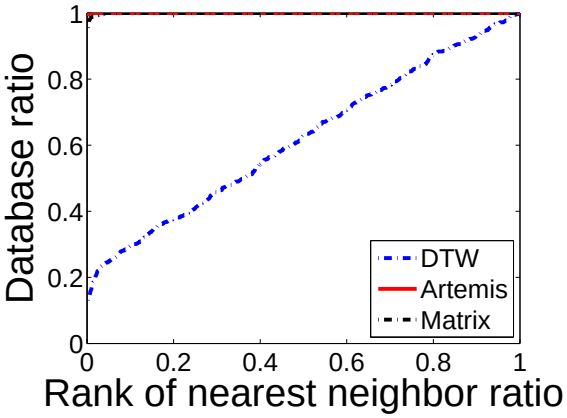


(c) Ranks of NN. Swap probability 1.0

Figure 6.4: Label swaps noise robustness experiment. ASL dataset.

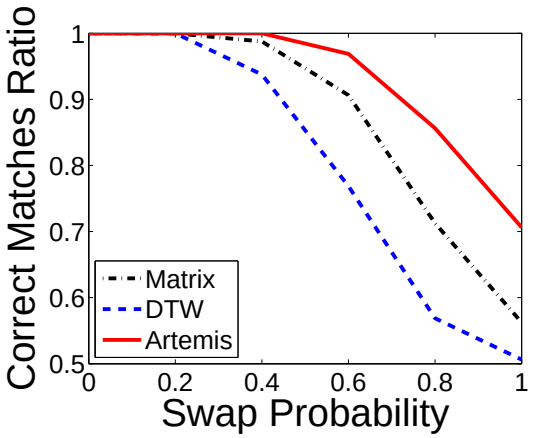


(a) Retrieval Accuracy.

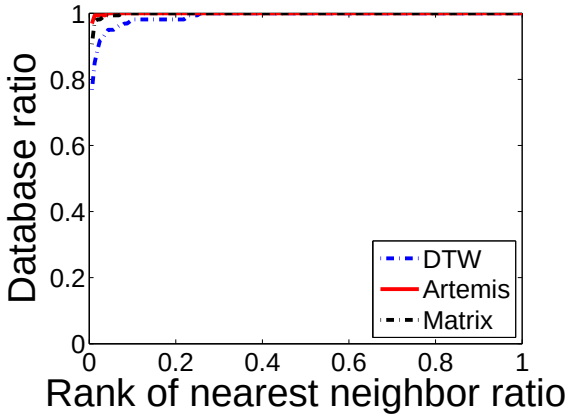


(b) Ranks of NN. Swap probability 1.0

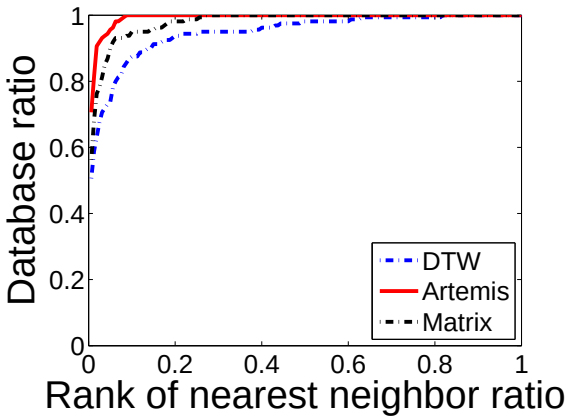
Figure 6.5: Label swaps noise robustness experiment. Hepatitis dataset.



(a) Retrieval Accuracy.

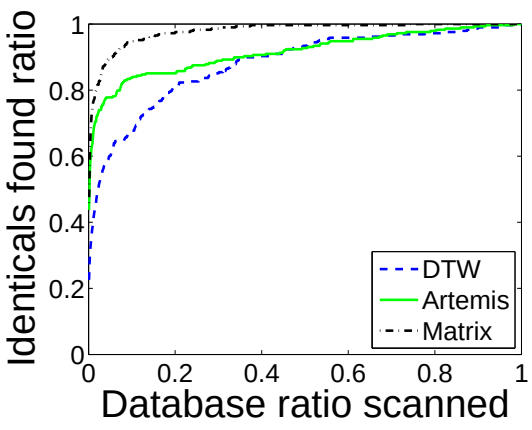


(b) Ranks of NN. Swap probability 0.6

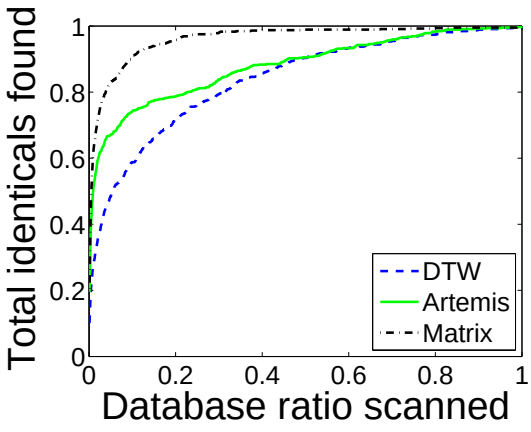


(c) Ranks of NN. Swap probability 1.0

Figure 6.6: Label swaps noise robustness experiment. Pioneer dataset.

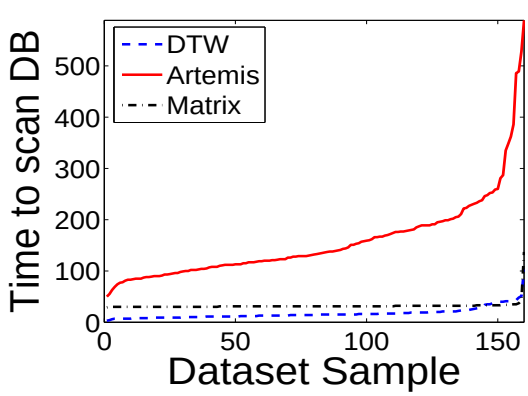


(a) Found identical phrases

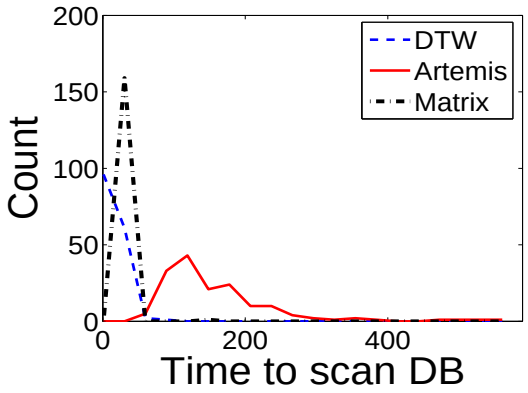


(b) Total sum of identical phrases found

Figure 6.7: Identical phrase experiment. ASL dataset.

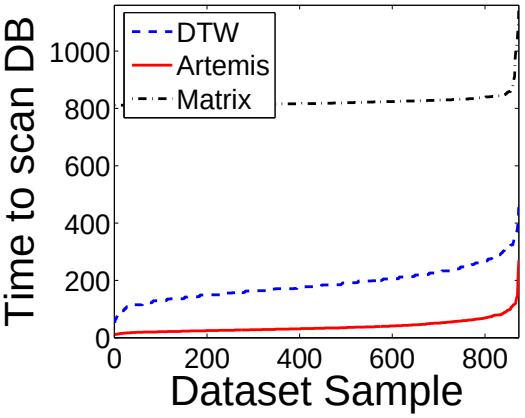


(a) Required times for each element, in ascending order.

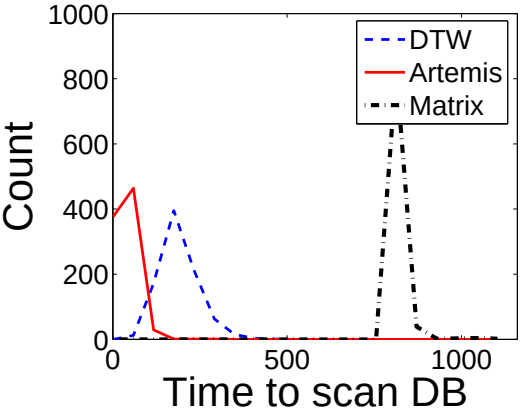


(b) Histograms of required times

Figure 6.8: Time (in Milliseconds) required to compare each elemet against the whole dataset. Pioneer dataset.

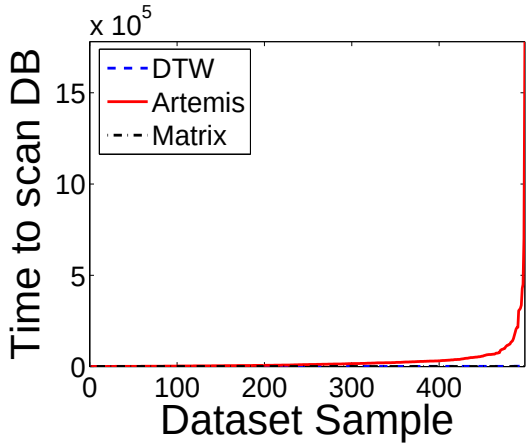


(a) Required times for each element, in ascending order.

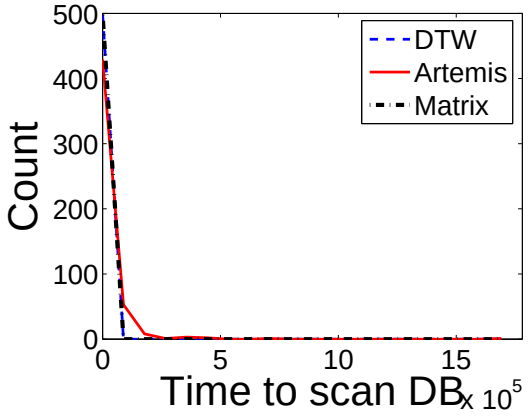


(b) Histograms of required times

Figure 6.9: Time (in Milliseconds) required to compare each elemet against the whole dataset. ASL dataset.



(a) Required times for each element, in ascending order.



(b) Histograms of required times

Figure 6.10: Time (in Milliseconds) required to compare each elemet against the whole dataset. Hepatitis dataset.

The assessment of $Artemis_{LB}$ is summarized in Table 6.3. The second column shows the tightness of the lower bounding technique while the third shows its pruning power. The higher values are observed on ASL, contrary to Pioneer which yields the lowest scores. The latter dataset consists of unequally represented classes; one of the three classes contains 102 out of 160 samples in total. E-sequences of the same class in the Pioneer dataset denote the same scenario in the robot's movement. Due to that, these e-sequences are expected to share large portions of common event-interval labels. Thus, they render $Artemis_{LB}$ unable to prune many of the comparisons.

Dataset	LB Tightness	1-NN pruning power
ASL	0.8837	0.7931
Hepatitis	0.7166	0.7012
Pioneer	0.6189	0.4855

Table 6.3: Lower Bound tightness and pruning power.

6.3 Lessons learned

The proposed distance functions were evaluated with respect to robustness against two types of artificial noise, k-NN classification accuracy, and scalability. In addition, the efficiency of the lower bound was tested by computing its tightness and its pruning power during 1-NN queries.

`Artemis` proved to be the most robust method to noise. The reason does not appear to be only the focus on the relations of the event-intervals, which is common with Relation Matrix. It is also the underlying nature of the method that attempts to find correspondence between event-intervals. The correspondence among event-intervals 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. On the other hand, the DTW approach examines the e-sequences point-by-point and the event-intervals out of their context. This makes it more sensitive to minor edit operations.

Consequently, DTW displayed a decline in performance in accordance with the increase of noise.

There was no clear winner, between the three proposed methods, in our k-NN classification experiments. The insight we acquired suggests that the choice must be application dependent. Additionally, the multiple possible ways that the class label can manifest in the e-sequences (e.g. as a single event-interval, a common sub-arrangement, a semantic combination) do not allow for a standard choice between our proposed methods.

The additional experiment on the ASL dataset supports the thesis that, if semantic information resides in the relations between event-intervals, the Relation Matrix measure has an advantage over the other two approaches. The reason is the fact that Relation Matrix assigns equal weight to all the event-interval relations and similar utterances in ASL share common pairs or event-interval relations. In fact, the insight we acquired suggests that in ASL common semantic information among similar utterances appears as sets of common sub-arrangements. In turn, this implies that the distance of pairs of e-sequences is not the most appropriate measure for studying ASL utterances.

The scalability experiments showed that DTW and Relation Matrix perform, in the average case, faster than `Artemis`, which suffers from a computational blow up for large e-sequences. In contrast, DTW and the Relation Matrix approach become slower when the alphabet size is significantly larger than the size of the e-sequences.

A way to overcome the increase in complexity due to the alphabet size would be to prune the alphabet during the comparisons and keep only the union of the labels present in the e-sequences that are being compared. That would yield a complexity of $O(|\mathcal{V}_\mathcal{S}||\mathcal{V}_\mathcal{T}| \cdot (|\mathcal{V}_\mathcal{S}| + |\mathcal{V}_\mathcal{T}|))$ for DTW and $O(|\mathcal{V}_\mathcal{S}|^2 + |\mathcal{V}_\mathcal{T}|^2 + (|\mathcal{V}_\mathcal{S}| + |\mathcal{V}_\mathcal{T}|)^2)$ for the Relation Matrix.

To speed up searches using `Artemis`, our lower bound technique $Artemis_{LB}$ proved significantly tight; the average tightness ranged from 61.8% (Pioneer dataset) up to 88% (ASL dataset). This translates to a pruning power of 48.5% to 79.3% over the brute force serial scan of the database. The fact that $Artemis_{LB}$ yields a zero distance score for pairs of e-sequences that share the same number of event-intervals for each label appears to affect only the experiments with the Pioneer dataset.

In conclusion, the choice of distance measure is application and task dependent. For clustering, the choice should depend on the way the classes are encoded into the e-sequences. For identifying exact similarity of e-sequences, `Artemis` is the only one that does not violate the identity of indiscernibles. Its high asymptotic complexity can be circumvented by the use of lower bounding techniques such as the one we presented.

Chapter 7

Summary and directions for Future Work

During the course of this thesis we studied the problem of comparing and analyzing arrangements of temporal intervals. We were the first to define the problem of comparing such arrangements [26] and provided three relevant distance functions: the Relation Matrix measure, the vector-based DTW and `Artemis`. In addition, we provided a linear-time lower-bounding technique for `Artemis`.

The three distance functions were tested on three datasets: American sign language, hepatitis and sensor data. The experiments included a thorough evaluation of the proposed distance functions in terms of robustness against two types of artificial noise, k-NN classification accuracy and scalability. `Artemis` and the Relation Matrix methods proved highly robust against noise. The results of k-NN classification were inconclusive as to which method should be preferred, while the scalability results were in accordance with the time complexity of the algorithms. Finally, the proposed lower bound proved to be highly tight and provided great pruning power for 1-NN search.

In addition, we studied the problem of finding the longest common sub-pattern in arrangements of temporal intervals. We proved that finding the LCSP of a pair of arrangements is an NP-complete problem. We presented an exact algorithm, based on dynamic programming, for finding the LCSP of a pair of arrangements. The algorithm is efficient for easy instances but requires exponential time for hard instances.

For the future, a series of further problems must be solved. Most importantly, `Artemis` must be further examined to determine if it obeys the triangular inequality. The result would answer the question whether `Artemis` is a metric. This would enable the creation of new and the use of existing techniques for indexing event-interval sequences. Similarly, additional lower bounding techniques, which provide greater tightness and pruning power, would facilitate the use of large event-interval sequence databases. For this setting, the bounding techniques must be either constant- or linear-time. Deriving metric lower bounds would be desirable in the effort to provide large-scale database services.

The proposed functions for comparing event-interval sequences require a priori knowledge of the whole sequences. Devising on-line algorithms [2], where the active arrangements are given in a streaming or bursty fashion, would provide solutions for additional real-world scenarios related to fields such as assistive environments. 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 [36] in an effort to provide faster solutions.

Randomized algorithms would prove useful for tackling hard instances of LCSP, too. In addition, approximation algorithms [18] are needed for many related computational tasks. It would be interesting to study the relation among hard instances of other NP-complete problems and instances of LCSP. Previous results regarding the hardness of approximating the Clique problem [12] provide the intuition that LCSP must be hard to approximate.

Furthermore, since we proved that event-interval sequences are an alternative to encoding multi-graphs with labeled edges, we must now study whether event-interval relations provide an additional scope for proving or determining properties in graphs and multi-graphs.

Summarizing our work on event-interval sequences, we solved the problem of comparing sequences. Existing work included mining the sequences for retrieving frequent patterns and association rules. Regarding event-interval sequences from the scope of data mining, it remains an open problem to find efficient algorithms for detecting closed patterns.

Bibliography

[1] T. Abraham and J. F. Roddick. Incremental meta-mining from large temporal data sets. In *ER '98: Proceedings of the Workshops on Data Warehousing and Data Mining*, pages 41–54, 1999.

[2] S. Albers. Online algorithms: a survey. *Mathematical Programming*, 97(1):3–26, 2003.

[3] J. M. Ale and G. H. Rossi. An approach to discovering temporal association rules. In *Proceedings of the 15th ACM Symposium On Applied Computing*, pages 294–300, 2000.

[4] J. Allen and G. Ferguson. Actions and events in interval temporal logic. *Journal of Logic and Computation*, 1994.

[5] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[6] B. Berendt. Explaining preferred mental models in Allen inferences with a metrical model of imagery. In *Proceedings of the 18th Annual Conference of the Cognitive Science Society*, pages 489–494, 1996.

[7] B. Bergen and N. Chang. Embodied construction grammar in simulation-based language understanding. *Construction grammars: Cognitive grounding and theoretical extensions*, pages 147–190, 2005.

[8] D. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *AAAI Workshop on Knowledge Discovery in Databases*, pages 359–370, 1994.

[9] G. Box and G. Jenkins. *Time Series Analysis: Forecasting and Control.* Prentice Hall PTR, 1994.

[10] X. Chen and I. Petrounias. Mining temporal features in association rules. In *Proceedings of the 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 295–300. Springer-Verlag, 1999.

[11] J. Edmonds and R. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, 1972.

[12] U. Feige, S. Goldwasser, L. Lovasz, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, pages 2–12. IEEE, 1991.

[13] F. Giannotti, M. Nanni, and D. Pedreschi. Efficient mining of temporally annotated sequences. In *Proceedings of the 6th SIAM Data Mining Conference*, volume 124, pages 348–359, 2006.

[14] F. Höppner. Discovery of temporal patterns - learning rules about the qualitative behaviour of time series. In *Proceedings of the 5th European Conference on Principles of Knowledge Discovery in Databases*, pages 192–203, 2001.

[15] F. Höppner and F. Klawonn. Finding informative rules in interval sequences. In *Proceedings of the 4th International Symposium on Advances in Intelligent Data Analysis*, pages 123–132, 2001.

[16] S.-Y. Hwang, C.-P. Wei, and W.-S. Yang. Discovery of temporal patterns from process instances. *Computers in Industry*, 53(3):345–364, 2004.

[17] F. Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 23(1):67–72, 1975.

[18] D. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, 1974.

[19] P. Kam and A. W. Fu. Discovering temporal patterns for interval-based events. In *Proceedings of the 2nd International Conference on Data Warehousing and Knowledge Discovery*, pages 317–326, 2000.

[20] R. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972.

[21] E. Keogh. Exact indexing of dynamic time warping. In *Proceedings of the 28th International Conference on Very Large Data Bases*, pages 406–417, 2002.

[22] E. Keogh and C. Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, 2005.

[23] S. Kim, S. Park, and W. Chu. An index-based approach for similarity search supporting time warping in large sequence databases. In *Proceedings of 17th International Conference on Data Engineering*, pages 607–614, 2001.

[24] R. Kosara and S. Miksch. Visualizing complex notions of time. *Studies in Health Technology and Informatics*, pages 211–215, 2001.

[25] O. Kostakis, P. Papapetrou, and J. Hollmén. Artemis: Assessing the similarity of event-interval sequences. In *Proceedings of the Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD 2011)*, volume 6912 of *Lecture Notes in Computer Science*, pages 229–244. Springer-Verlag, 2011.

[26] O. Kostakis, P. Papapetrou, and J. Hollmén. Distance measure for querying arrangements of temporal intervals. In *Proceedings of Pervasive Technologies Related to Assistive Environments*, 2011.

[27] J. Kruskal and M. Liberman. The symmetric time warping problem: From continuous to discrete. *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*, pages 125–161, 1983.

[28] S. Laxman, P. Sastry, and K. Unnikrishnan. Discovering frequent generalized episodes when events persist for different durations. *IEEE Transactions on Knowledge and Data Engineering*, 19(9):1188–1201, 2007.

[29] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics*, 10(8):707–710, 1966.

[30] J.-L. Lin. Mining maximal frequent intervals. In *Proceedings of the 18th ACM Symposium On Applied Computing*, pages 624–629.

[31] H. Lu, J. Han, and L. Feng. Stock movement precdiction and n-dimensional intertransaction association rules. In *Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 12:1–7, 1998.

[32] C. Mooney and J. F. Roddick. Mining relationships between interacting episodes. In *Proceedings of the 4th SIAM International Conference on Data Mining*, 2004.

[33] F. Mörchen. Unsupervised pattern mining from symbolic temporal data. *SIGKDD Exploration Newsletter*, 9:41–55, June 2007.

[34] F. Mörchen. 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*, KDD '10, pages 2:1–2:1. ACM, 2010.

[35] F. Mörchen and D. Fradkin. Robust mining of time intervals with semi-interval partial order patterns. In *Proceedings of the 10th SIAM International Conference on Data Mining*, pages 315–326, 2010.

[36] R. Motwani and P. Raghavan. Randomized algorithms. *ACM Computing Surveys*, 28(1):33–37, 1996.

[37] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[38] F. Pachet, G. Ramalho, and J. Carrive. Representing temporal musical objects and reasoning in the MusES system. *Journal of New Music Research*, 25(3):252–275, 1996.

[39] P. Papapetrou, G. Benson, and G. Kollios. Discovering frequent poly-regions in DNA sequences. In *Proceedings of the IEEE ICDM Workshop on Data Mining in Bioinformatics*, 2006.

[40] P. Papapetrou, G. Kollios, S. Sclaroff, and D. Gunopulos. Discovering frequent arrangements of temporal intervals. In *Proceedings of 5th IEEE International Conference on Data Mining*, pages 354–361, 2005.

[41] P. Papapetrou, G. Kollios, S. Sclaroff, and D. Gunopulos. Mining frequent arrangements of temporal intervals. *Knowledge and Information Systems*, pages 133–171, 2009.

[42] D. Patel, W. Hsu, and M. Lee. Mining relationships among interval-based events for classification. In *Proceedings of the 28th ACM SIGMOD International Conference on Management of Data*, pages 393–404. ACM, 2008.

[43] N. Pissinou, I. Radev, and K. Makki. Spatio-temporal modeling in video and multimedia geographic information systems. *GeoInformatica*, 5(4):375–409, 2001.

[44] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):43–49, 1978.

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

[46] R. Villafane, K. A. Hua, D. Tran, and B. Maulik. Knowledge discovery from series of interval events. *Intelligent Information Systems*, 15(1):71–89, 2000.

[47] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh. Indexing multidimensional time-series. *The VLDB Journal*, 15:1–20, January 2006.

[48] E. Winarko and J. F. Roddick. Armada - an algorithm for discovering richer relative temporal association rules from interval-based data. *Data & Knowledge Engineering*, 63(1):76–90, 2007.

[49] S.-Y. Wu and Y.-L. Chen. Mining nonambiguous temporal patterns for interval-based events. *IEEE Transactions on Knowledge and Data Engineering*, 19(6):742–758, 2007.

[50] B. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary $\mathcal{L}_p$ norms. In *Proceedings of the 26th International Conference on Very Large Data Bases*, pages 385–394. Citeseer, 2000.