Efficient Nearest Neighbor Search on Metric Time Series

Dissertation

zur Erlangung des akademischen Grades
doctor rerum naturalium

im Fach Informatik

eingereicht an der Mathematisch–Naturwissenschaftlichen Fakultät der
Humboldt–Universität zu Berlin
von

Dipl. Inf. Jörg Peter Schäfer (geb. Bachmann)

Präsidentin der Humboldt–Universität zu Berlin
Prof. Dr. Peter Frensch, (komm.) Präsident der Humboldt-Universität zu Berlin

Dekan der Mathematisch–Naturwissenschaftlichen Fakultät
Prof. Dr. Caren Tischendorf

1. Gutachterin:	Prof. Dr. Nicole Schweikardt
2. Gutachter:	Prof. Dr. Ulf Leser
3. Gutachter:	Prof. Themis Palpanas, PhD

Tag der mündlichen Prüfung: 22.04.2022

Abstract

While Deep-Learning approaches beat Nearest-Neighbor classifiers in an increasing number of areas, searching existing uncertain data remains an exclusive task for similarity search. Numerous specific solutions exist for different types of data and queries. This thesis aims at finding fast and general solutions for searching and indexing arbitrarily typed time series.

A time series is considered a sequence of elements where the elements' order matters but not their actual time stamps. Since this thesis focuses on *measuring distances* between time series, the metric space is the most appropriate concept where the time series' elements come from. Hence, this thesis mainly considers *metric time series* as data type. Simple examples include time series in Euclidean vector spaces or graphs.

For general similarity search solutions in time series, two primitive comparison semantics need to be distinguished, the first of which compares the time series' trajectories ignoring time warping. A ubiquitous example of such a distance function is the Dynamic Time Warping distance (`DTW`) developed in the area of speech recognition. The Dog Keeper distance (`DK`) is another time-warping distance that, opposed to `DTW`, is truly invariant under time warping and yields a metric space. After canonically extending `DTW` to accept multi-dimensional time series, this thesis contributes a new algorithm computing `DK` that outperforms `DTW` on time series in high-dimensional vector spaces by more than one order of magnitude. An analytical study of both distance functions reveals the reasons for the superiority of `DK` over `DTW` in high-dimensional spaces.

The second comparison semantic compares time series in Euclidean vector spaces regardless of their position or orientation. This thesis proposes the Congruence distance that is the Euclidean distance minimized under all isometric transformations; thus, it is invariant under translation, rotation, and reflection of the time series and therefore disregards the position or orientation of the time series. A proof contributed in this thesis shows that there can be no efficient algorithm computing this distance function (unless P=NP). Therefore, this thesis contributes the Delta distance, a metric distance function serving as a lower bound for the Congruence distance. While the Delta distance has quadratic time complexity, the provided evaluation shows a speedup of more than two orders of magnitude against the Con-

gruence distance. Furthermore, the Delta distance is shown to be tight on random time series, although the tightness can be arbitrarily bad in corner-case situations.

Orthogonally to the previous mentioned comparison semantics, similarity search on time series consists of two different types of queries: *whole sequence matching* and *subsequence* search. Metric index structures (e. g., the M-Tree) only provide whole matching queries natively. This thesis contributes the concept of metric subset spaces and the SuperM-Tree for indexing metric subset spaces as a generic solution for subsequence search. Examples for metric subset spaces include subsequence search regarding the distance functions from the comparison semantics mentioned above. The provided evaluation shows that the SuperM-Tree outperforms a linear search by multiple orders of magnitude.

Acknowledgements

I am deeply grateful to my supervisor Prof. Dr. Nicole Schweikardt for her support, insightful comments, and suggestions, especially on one of the biggest chapters in this thesis, as well as for her patience and motivation. I am also very grateful to my supervisor Prof. Johann-Christoph Freytag, Ph. D. for his patience and support to let me research in an area that was entirely new for him, which I imagine challenging. Louchka Popova-Zeugmann supported me throughout all my studies. She opened up the opportunity to this thesis in the first place.

I would also like to thank all members of the Humboldt–Universität zu Berlin. It was a helpful, friendly, and sometimes even familiar environment that I enjoyed every day. Sincere thanks also go to my colleagues, especially Matthias J. Sax, Jochen Taeschner, and Benjamin Hauskeller. It was their interest and support that helped me structure the work and kept me motivated. I appreciate a lot the intense work and patience of my proofreaders Matthias J. Sax, Thomas Janda, and Andreas Leich.

Without the patience and the cheering up of my new colleagues at the DLR, I wouldn't have been able to finish this work. Finally, I would like to express my sincere gratitude to my beloved family. My wife continuously showed tremendous understanding and encouragement during the entire time. My son's understanding is not taken for granted when I couldn't spend the time playing with him. The continuous smiling of my newborn daughter encouraged me, especially in exhausting times.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Section 1.1 provides an intuitive understanding of the topics and issues addressed in this thesis. Section 1.2 briefly describes some related work. The contributions and cooperations regarding these contributions are declared in Section 1.3 and 1.4, respectively. Section 1.5 outlines this thesis.

1.1 Motivation

Time-warping invariant distance: Many multimedia retrieval applications require to find similar time series to a given query time series [41,66,72]. One common application of finding similar time series is multimedia retrieval, including motion gesture recognition, speech recognition, and classification of handwritten letters. All these tasks have in common that the time series of the same classes (e. g., same spoken words or same gestures) follow a similar path in space but have some temporal distortion. Another example is tracking the GPS coordinates of two cars driving the same route. If these two time series shall be recognized as similar, temporal differences resulting from driving style, traffic lights, and traffic jams need to be dealt with. *Time-warping distance functions* considered in Chapter 3 (e. g., Dynamic Time Warping (`DTW`) [80], edit distance with real penalties (`ERP`) [33], and the Dog Keeper distance (`DK`) [9, 10, 40]) respect this semantic requirement.

Isometric invariant distance: Time series of the same class (e. g., same written characters or same gestures) may differ by temporal and spatial displacements [59]. While time-warping distance functions are robust against temporal displacements, they generally fail when the time series are rotated or translated in space. Distance functions that measure the (approximate) congruence of two time series are defined and analyzed in Chapter 4. Thereby, the distance between two time series S and T shall be 0 iff S can be transformed into T by rotation, translation, and mirroring; in this case,

S and T are said to be *congruent*. A value greater than 0 shall correlate to the amount of transformation needed to turn the time series into congruent ones.

Query types: While some applications compare whole time series against each other, other applications need to search for shorter sequences within long time series, e. g., when searching for a specific pathological sample within a long-term ECG[1]. These tasks are refered to as *whole sequence matching* and *subsequence matching*, respectively [41].

Indexing: Notably, this thesis takes into account subsequence search on multi-dimensional time series using time warping and congruence distance functions, respectively. Since datasets increase in size, computational performance remains crucial when designing new algorithms and data structures for these tasks. To improve the runtime performance, one can improve the computation time of the comparison function and reduce the number of overall comparisons. Two common approaches for pruning comparisons are lower bounds for the distance function and index structures.

Elegant solutions exist for index structures providing range queries on ordered data (e. g., the B^+-Tree [36]), range queries on spatial data in vector spaces (e. g., the R^*-Tree [16]), and range queries on data in metric spaces (e. g., the M-Tree [35]). In particular, metric index structures elegantly separate the semantics of the comparison and the indexing technique [48]: The algorithms of the index structure only depend on specific properties of the distance function. Hence, the program code for the index structure and the distance function can be maintained separately.

Problem statement and solution approach: Unfortunately, computing time-warping distance functions seems to have quadratic time complexity [15, 25, 26]. Chapter 3 even provides proof that the runtime of similarity search applications using `DTW` strongly suffers under the curse of dimensionality, i. e., the runtime decreases drastically with increasing dimensionality. This thesis addresses these performance issues by providing fast algorithms for the `DK` distance that yield similar quality in classification tasks [9].

When measuring the degree of congruence of two time series, only rotation on two-dimensional time series has been considered [5, 59]. Chapter 4 discusses measuring the congruence on time series with arbitrary dimensionality regarding arbitrary isometric transformations (i. e., the transformation moves the time series by translation, rotation, and reflection).

Since metric distance functions are not applicable for subsequence search, Chapter 5 fills this gap by introducing *metric subset spaces*. The corresponding *subset distance functions* are naturally designed for containment

[1] An electrocardiogram (short ECG) shows the electrical activity of the heart over time.

comparison (in particular subsequence search). The Euclidean distance, the Dog Keeper distance, and even a distance function on sets of metric elements (i. e., the Hausdorff distance) can be canonically extended such that they yield metric subset spaces. Furthermore, Chapter 5 contributes the SuperM-Tree that is an index structure for any metric subset space, and thus achieves the same separability as metric index structures for metric spaces.

1.2 Related Work

Similarity search is used in various ways, some of which describe the relation of two objects, and some of which solve classification or regression tasks. Examples relating two objects to each other include content-based video copy detection (CBCD) and cover song recognition. CBCD applications, for example, search for similar videos that originate from the same source while transformations such as resolution change or fast-forwarding might have changed the video [66,86]. On the other hand, Song cover recognition allows similar songs to be from different sources, but they shall have similar melody and arrangements [37,54]. Possible implementations for both applications use distance functions to perform similarity search on a set of features describing the data objects. Predicting an answer to a query by using the labels of objects that are similar to the query object is a common way to implement simple classifiers or regressors (e. g., text categorization [21] and melting point prediction [75], respectively). Despite the numerous similarity search applications, this thesis focuses on fundamental research on similarity search concepts, while in-depth studies of its applications are out of scope for this thesis.

Time series representations: In the area of similarity search on real-valued time series, much research aims to improve the algorithm's runtime by using different (mostly approximating) representations of the time series. Some of the approaches reduce the representational dimensionality, e. g., the Piecewise Linear Approximation (PLA) [34], or the Singular Value Decomposition (SVD) [62]. PLA reduces the dimensionality by averaging successive elements (i. e., by reducing the length of the time series). SVD considers a set of one-dimensional time series as a huge matrix, performs the singular value decomposition, and then ignores all but the most dominant Eigenvectors. Other representational changes transform the time series into their frequency domain, e. g., the Discrete Fourier Transform (DFT) [1, 42], the Discrete Wavelet Transform (DWT) [30,61], or the Symbolic Fourier Approximation (SFA) [81]. The latter compresses the Fourier coefficients using symbolic constants for ranges of values, which opens a third category of representational changes. The earliest published approach that proposed such

symbolic constants is the Symbolic Aggregate approXimation (SAX) [68,69]. Numerous further compression and dimensionality reduction techniques exist. However, despite their success, they consider mainly one-dimensional time series, which is a severe limitation for this thesis' goals. Hence, this thesis does not study the optimization of a query's runtime by changing the time series' representation.

Time-warping distance functions: Numerous distance functions exist for the uncountable applications that yield different semantic requirements to the distance function [38, 41]. Time-warping distance functions consider distortion in time, e. g., Dynamic Time Warping (DTW) [80], the Dog Keeper distance (DK) [43], and the edit distance with real penalties (ERP) [33].[2] While `DTW` is the most famous time-warping distance function, the `DK` distance is the oldest one, presented by M. Fréchet in 1906. Regarding their semantic, the difference is analogous to the difference between the Euclidean norm and the Maximum norm of n-dimensional real-valued vectors.

Since the runtime complexity for several time-warping distance functions is proven to be quadratic [15, 25, 26], some approaches aim at reducing the expected computation time [3, 9, 14], other approaches aim at improving the runtime using approximations. For example, the restriction of time-warping to a maximum distance in time improves the runtime by a constant factor (e. g., the Sakoe-Chiba band [80] or the Itakura parallelogram [51]); cheap to compute lower bounds to the distance function allow pruning the computation of the expensive, actual distance function if the lower bound promises a value beyond a certain threshold [58, 60, 64, 88]; the time series' representational changes mentioned above improve the runtime at the cost of approximated distances.

Instead of proposing a new time-warping aware distance function or a lower bound for one of the distance functions, this thesis presents a close study and comparison of `DTW` vs. `DK`. This thesis does, however, propose *Sparse Dog Keeper*, a new algorithm for the computation of the `DK` distance.

Congruence of time series: Some applications require distance functions that measure the congruence of two time series. For example, Keogh et al. adapted `DTW` to approximate the congruence of two shapes [59] by creating a time series per shape that measures the distance of the shape's center to the furthest point of the shape in each direction, respectively; these time series are then compared using `DTW`. While this approach works well for this specific application, it does not apply to high-dimensional time series and arbitrary isometric transformations. Alt et al. categorized groups of congruence transformations; they distinguished between translation, rotation,

[2] Details on these distance functions are presented in Section 2.4.1.

reflection, scaling, reordering of the points, and combinations of those [5]. They further identified different tasks, i. e., whether the allowed error for the approximation is already given or to be determined; in the former case, the congruence problem becomes a decision problem. Various studies considered two- and three-dimensional spaces: Atkinson proposed an algorithm that solved the congruence problem in quasi-linear time on sets of three-dimensional points [7]; Heffernan and Schirra considered congruence problems in two-dimensional spaces [45]; Indyk and Venkatasubramanian even achieved a near-linear complexity algorithm by applying of Hall's Theorem; their distance function, however, relaxes the one-to-one condition of the mapping [50]. Cabello, Giannopolos, and Knauer showed that the exact congruence problem on unordered point sets is equivalent to graph isomorphism [27]. Akutsu provided a randomized algorithm for the congruence problem in arbitrary Euclidean spaces that is exponential in the dimensionality and polynomial in the cardinality of the sets [2]. This thesis studies the congruence of ordered point sets with unbound dimensionality regarding arbitrary isometric transformations. Moreover, this thesis provides proof that the decision of the approximated congruence problem under these circumstances already is NP-hard.

Index structures for complex data types: Numerous index structures exist offering queries on complex data types such as time series, including index structures for sets and set containment joins [53, 63, 87], for strings and similarity search regarding the Edit distance [84], for one-dimensional time series and Dynamic Time Warping [28, 57, 58]. These index structures are, however, domain-specific solutions or at least specific to fixed distance functions. On the other hand, metric index structures that support nearest neighbor queries in any metric space [19, 23, 35, 76] are more generic and support range queries on a wide range of different complex data types. These index structures use the triangle inequality to estimate distances and prune parts of the indexed dataset. This thesis is inspired by the ideas of metric spaces and metric index structures and enhances them to enable new query types for metric time series.

Index structures supporting subsequence queries: Indexing time series and supporting subsequence queries is even more challenging. Faloutsos, Ranganathan, and Manolopoulos propose the ST-index for subsequence search regarding the Euclidean distance [42]. For each indexed time series, they store all subsequences' two most dominant Fourier coefficients in an R*-Tree [16]. At query time, they exploit that the Euclidean distance of such Fourier coefficients is a lower bound for the actual Euclidean distance of the time series; thus, a range query on the R*-Tree returns a set of candidate time series. Rakthanmanon et al. built an algorithm for searching huge

amounts of time series; they support subsequence queries regarding DTW [78]. Their main runtime improvement results from a well-chosen cascade of lower bounds to DTW. Bhaduri et al. improve the runtime of subsequence queries by exploiting the triangle inequality on subsequences within a large time series and pruning candidate positions within the large time series, i. e., they built an index structure that holds a set of subsequences that are all of the same lengths [20]. This thesis proposes the SuperM-Tree, an index structure for subsequence search that is as generic as any metric index structure.

1.3 Contributions

The following contributions originate from this work:

- two dataset generators for multi-dimensional time series (RAM and the multi-dimensional CBF) applicable for benchmarking time-warping and congruence distance functions;
- an analytical and experimental evaluation of $\mathtt{LB}_{\mathtt{Box}}$ (a generalization of Keogh's lower bound for DTW to multi-dimensional time series) regarding the curse of dimensionality;
- a proof in modern mathematical notation showing that the Dog Keeper distance (DK) satisfies the triangle inequality[3];
- a fast algorithm for DK and an experimental evaluation and comparison of DK against DTW and $\mathtt{LB}_{\mathtt{Box}}$;
- a proof showing that there is no fast algorithm for the congruence distance function unless the complexity classes P and NP are equal;
- a fast approximating algorithm for the congruence distance;
- the concept of metric subset spaces including corresponding indexing techniques;
- an implementation of an index structure for metric subset spaces (SuperM-Tree), including an experimental evaluation.

1.4 Cooperations

Most of the contributions presented in this thesis are public already. The following list references these publications and circumscribes the part of the work of the co-authors.

[3] A different proof by Maurice Fréchet exists [43]. However, it is written in french and it uses a hard to read mathematical notation.

- The time series generators described in Section 2.5 have been published in [12]. The co-author J.-C. Freytag supervised the work.
- The proof of DK's triangle inequality (see Section 2.4) is published in [9]. The co-author J.-C. Freytag supervised the work.
- The contents of Chapter 3 have been published in [9] and [10]. The co-author J.-C. Freytag supervised the work.
- The definition of the Congruence distance in Chapter 4 and the proof of its hard computation have been published in [13]. The co-authors J.-C. Freytag and Nicole Schweikardt supervised the work. Nicole Schweikardt further found a mistake in the first draft of the proof. Jörg P. Schäfer, the author of this thesis, corrected the proof before publishing [13]. Benjamin Hauskeller helped to implement and run the experiments.
- The approximating algorithms in Chapter 4 have been published in [11]. The co-author J.-C. Freytag supervised the work.
- The contents of Chapter 5 have been published in [8].

1.5 Outline

The rest of this thesis is structured as follows: Chapter 2 introduces the main work with basic notations in Section 2.1, preliminaries to metric spaces and metric index structures in Section 2.2, and time series, distance functions on time series, and time series datasets and dataset generators in Section 2.3, 2.4, and 2.5, respectively.

Chapter 3 discusses time-warping distance functions. After describing specific nomenclature in Section 3.2, Section 3.3 introduces $\mathtt{LB}_{\mathtt{Box}}$ as canonical extension to Keogh's lower bound on DTW for multi-dimensional time series. Section 3.3 further shows that $\mathtt{LB}_{\mathtt{Box}}$ suffers from the curse of dimensionality. Section 3.4 provides alternative algorithms for the DK distance. The algorithms provided in Section 3.3 and 3.4 are compared and evaluated in Section 3.5.

Chapter 4 discusses distance functions considering congruence of time series. After motivating this topic and the problem statement in Section 4.1, the Congruence distance is introduced in Section 4.2. Moreover, Section 4.2 shows that any possible algorithm computing the Congruence distance up to a certain precision has at least exponential runtime unless the complexity classes P and NP are equal. Various approximating algorithms with different properties are discussed in Section 4.3 and evaluated in Section 4.4.

Chapter 5 presents a new index method for subsequence search. After introducing to the topic and the approach followed in this chapter in

Section 5.1, Section 5.2 presents the new formal concept of metric subset spaces as well as three basic examples which cover the distance functions from Chapter 3 and 4. Section 5.3 describes the SuperMTree, which is an index structure for metric subset spaces; Section 5.4 presents empirical results on the three former examples.

Chapter 6 concludes this thesis.

Chapter 2

Preliminaries

This chapter clarifies the basic notation, nomenclature, and concepts used throughout the rest of this work. This chapter also describes the dataset generators used in the experiments in the following chapters.

2.1 Notation

Sets of numbers: Sets are denoted using brackets, i. e., $\{x, y, z\}$ is a set of three elements. For a set A, the boolean term $x \in A$ is true, iff x is an element in A. For two sets A and B, $A \cup B$, $A \cap B$, and $A \setminus B$ denote the union, the intersection, and the difference of both sets, respectively. The set of non-negative integers, the set of reals, and the set of all reals $\geqslant c$, for some $c \in \mathbb{R}$ are denoted by $\mathbb{N}$, $\mathbb{R}$, and $\mathbb{R}_{\geqslant c}$, respectively. The cardinality of a set A is denoted by $\#A$. Having a numeric function f, $\arg\min\{f(x) \mid x \in M\}$ returns an argument x which minimizes the function f. For $a, b \in \mathbb{R}$, $a \approx b$ denotes that they are equal, approximately.

Logic: For two boolean variables F, G the logical *and* and *or* are denoted as $F \wedge G$ and $F \vee G$, respectively. The logical implication is denoted by $\Longrightarrow$. The logical *and* and *or* of a set of boolean variables $F_1, \ldots, F_n$ is denoted by $\bigwedge_{1 \leqslant i \leqslant n} F_i \coloneqq F_1 \wedge \cdots \wedge F_n$ and $\bigvee_{1 \leqslant i \leqslant n} F_i \coloneqq F_1 \vee \cdots \vee F_n$, respectively. The all quantor is denoted by $\forall$, and the exists quantor is denoted by $\exists$. For a parametrized boolean term F, the term $\exists x \exists y F(x, y)$ is abbreviated with $\exists x, y : F(x, y)$. The term $\exists x \in A : F(x)$ is written instead of $\exists x : x \in A \Rightarrow F(x)$ for better readability.

Mappings: For mappings $f : A \longrightarrow B$ and $g : B \longrightarrow C$, $f(A) \coloneqq \{f(x) \mid x \in A\}$ denotes the image of f and $g \circ f : x \mapsto g(f(x))$ denotes the concatenation of g and f. Furthermore, $\inf f$ and $\sup f$ are the infimum and the supremum of $f(A)$, respectively. The absoloute mapping $|\cdot| : \mathbb{R} \longrightarrow \mathbb{R}_{\geqslant 0}$ is defined by $|x| \coloneqq \max\{x, -x\}$.

Intervals: For numbers x, y the interval consisting of all *numbers* z with $x \leqslant z \leqslant y$ is denoted by $[x, y]$. Furthermore, $[x, y) \coloneqq [x, y] \setminus \{y\}$. If $I = [x, y]$ declares an index set, it is restricted to the set of *integers*, e. g., for a given set $S = \{s_1, \cdots, s_9\}$ and an interval $[2, 4]$, it is $I = \{2, 3, 4\}$ and $S_I = \{s_2, s_3, s_4\}$. The d-dimensional cartesian product is denoted using the symbol $\bigotimes$, thus $\bigotimes_{1 \leqslant j \leqslant d} [l_i, u_i]$ denotes the set of vectors v with $l_j \leqslant v_j \leqslant u_j$ for $1 \leqslant j \leqslant d$.

Vector spaces: For $k \in \mathbb{N}$, the set of all vectors of length k is denoted by $\mathbb{R}^k$. For a vector $v \in \mathbb{R}^k$, the term v_i denotes its entry at position i. The *i-th unit vector* in $\mathbb{R}^k$ is denoted by e_i, i.e., the vector with entry 1 in the i-th position and entry 0 in all other positions. The usual *scalar product* on $\mathbb{R}^k$ is denoted by $\langle \cdot, \cdot \rangle$; i.e., $\langle u, v \rangle = \sum_{i=1}^{k} u_i v_i$ for $u, v \in \mathbb{R}^k$.

Vector norms: For $p \in \mathbb{R}_{\geqslant 1}$ the usual *p-norm* on $\mathbb{R}^k$ is denoted by $\|\cdot\|_p$; i.e., $\|v\|_p = \left(\sum_{i=1}^{k} |v_i|^p\right)^{1/p}$ for all $v \in \mathbb{R}^k$. In particular, $\|\cdot\|_1$ denotes the *Manhatten norm*, $\|\cdot\|_2$ denotes the *Euclidean norm*, and $\|\cdot\|_\infty$ denotes the *Maximum norm*. Note, that $\|v\|_2 = \sqrt{\langle v, v \rangle}$ for all $v \in \mathbb{R}^k$.

In general, a *vector norm* is an arbitrary mapping $\|\cdot\| : \mathbb{R}^k \longrightarrow \mathbb{R}_{\geqslant 0}$ that satisfies the following axioms:

$$\begin{aligned} \forall\ v \in \mathbb{R}^k &: \ \|v\| = 0 \implies v = 0. \\ \forall\ \lambda \in \mathbb{R},\ v \in \mathbb{R}^k &: \ \|\lambda v\| = |\lambda| \cdot \|v\|. \\ \forall\ u, v \in \mathbb{R}^k &: \ \|u + v\| \leqslant \|u\| + \|v\|. \end{aligned}$$

Matrices: An $m \times n$ matrix is denoted by $A = (a_{i,j})$. The set of all $(k \times k)$-matrices with entries in $\mathbb{R}$ is denoted by $\mathbb{R}^{k \times k}$, for $k \in \mathbb{N}$. Given a matrix $A \in \mathbb{R}^{k \times k}$, $A_{i,j}$ denotes the element in the i-th row and j-th column. A matrix $M \in \mathbb{R}^{k \times k}$ is called *orthogonal* if the absolute value of its determinant is 1. Equivalently, M is orthogonal iff $\langle m_i, m_i \rangle = 1$ and $\langle m_i, m_j \rangle = 0$ for all $i, j \in [1, k]$ with $i \neq j$, where m_i denotes the vector in the i-th column of M. The set of all orthogonal matrices in $\mathbb{R}^{k \times k}$ is denoted by $\mathcal{MO}(k)$. The product of the matrix $M \in \mathbb{R}^{k \times k}$ and the vector $v \in \mathbb{R}^k$ is denoted by Mv and $M \cdot v$. The terms λv, $\lambda \cdot v$, λM, and $\lambda \cdot M$ denotes the product of the number $\lambda \in \mathbb{R}$ with the vector v and the matrix M, respectively. Recall that two vectors $u, v \in \mathbb{R}^k$ are orthogonal iff $\langle u, v \rangle = 0$ and that angles and lengths are invariant under multiplication with orthogonal matrices, i. e.:

$$\begin{aligned} \forall\ u, v \in \mathbb{R}^k,\ M \in \mathcal{MO}(k) &: \ \langle Mu, Mv \rangle = \langle u, v \rangle. \\ \forall\ u \in \mathbb{R}^k,\ M \in \mathcal{MO}(k) &: \ \|Mu\|_2 = \|u\|_2. \end{aligned}$$

Matrix norms: A *matrix norm* is a mapping $\|\cdot\| : \mathbb{R}^{k\times k} \longrightarrow \mathbb{R}_{\geqslant 0}$ satisfying the following axioms:

$$\begin{aligned} \forall\, M \in \mathbb{R}^{k\times k} \;&:\; \|M\| = 0 \implies M = 0. \\ \forall\, \lambda \in \mathbb{R},\ M \in \mathbb{R}^{k\times k} \;&:\; \|\lambda M\| = |\lambda| \cdot \|M\|. \\ \forall\, M, M' \in \mathbb{R}^{k\times k} \;&:\; \|M + M'\| \leqslant \|M\| + \|M'\|. \end{aligned}$$

The particular matrix norms considered in this thesis are the *max column norm* $\|\cdot\|_m$ and the *p-norm* $\|\cdot\|_p$, for $p \in \mathbb{R}_{\geqslant 1}$, which are defined as follows: For all $M \in \mathbb{R}^{k\times k}$,

$$\begin{aligned} \|M\|_m &:= \max_{j\in[1,k]} \Big(\sum_{i=1}^{k} |m_{i,j}| \Big), \\ \|M\|_p &:= \Big(\sum_{i=1}^{k} \sum_{j=1}^{k} |m_{i,j}|^p \Big)^{1/p}. \end{aligned}$$

Sequences: Sequences (e. g., time series) are usually written using capital letters, e. g., $S = (s_1, \cdots, s_n)$ is a sequence of length n. The length of a finite sequence S is denoted by $\#S$. Suppose $s_i \in \mathbb{R}^d$, then $s_{i,j}$ denotes the j-th element of the i-th vector in the sequence S. The projection to the j-th dimension is denoted by S^j, i. e., $S^j = (s_{1,j}, \cdots, s_{n,j})$. For a set $\mathbb{M}$, the term $\mathbb{M}^{\mathbb{N}}$ denotes for the union $\bigcup_{n\in\mathbb{N}} \mathbb{M}^n$. Hence, $\mathbb{M}^{\mathbb{N}}$ denotes the set of sequences over $\mathbb{M}$ of arbitrary lengths. For an infinite sequence $S = (s_i)_{i\in\mathbb{N}}$, the limit is denoted by $S \longrightarrow s^*$ for $i \to \infty$.

Operations on sets and sequences: If there are no ambiguities, operations on sets and sequences apply elementwise: $f(S) \coloneqq \{f(s_1), \cdots, f(s_n)\}$ for a set $S \subset \mathbb{M}$ and a function on $\mathbb{M}$ and $f(T) \coloneqq (f(t_1), \cdots, f(t_n))$ for a sequence in $\mathbb{M}$ and a function on $\mathbb{M}$, e. g., $|S| = \{|s_1|, \cdots, |s_n|\}$ and $\sqrt{T} = \big(\sqrt{t_1}, \cdots, \sqrt{t_n}\big)$. Analogously, binary operations apply elementwise, i. e., $M \cdot S = \{Ms_1, \cdots, Ms_n\}$, $S + v = \{s_1 + v, \cdots, s_n + v\}$, $M \cdot T = (M \cdot t_1, \cdots, M \cdot t_n)$, and $T + v = (t_1 + v, \cdots, t_n + v)$ for a set $S = \{s_1, \cdots, s_n\}$, a sequence $T = (t_1, \cdots, t_n)$ in $\mathbb{R}^k$, a matrix $M \in \mathbb{R}^{k\times k}$, and a vector $v \in \mathbb{R}^k$, respectively.

Random variables: For a random variable X over $\mathbb{R}$, its mean is denoted by $\mathbb{E}[X]$, its variance is denoted by $\mathbb{V}[X]$, and the probability measure is denoted by $\mathbb{P}$, i. e., $\mathbb{P}[X < a]$ is the probability that the value of X is less than $a \in \mathbb{R}$. The standard normal distribution is denoted by $\Phi_{0,1}$.

Algorithms: Abbreviations of algorithms are usually written in capital letters (e. g., `DTW` for Dynamic Time Warping or `CBF` for Cylinder-Bell-Funnel generator). Helper functions for algorithms are written in mono space fonts (e. g., `helperFunction`).

2.2 Similarity search and metric spaces

Similarity search or nearest neighbor search is a common problem in computer science [41, 49, 70]. For a given dataset and query, the problem is to find the nearest neighbor to the query in the dataset regarding a certain distance or similarity function. The difference between distance and similarity functions is that a distance function returns 0 for exact matches and a higher value otherwise, whereas similarity functions return larger values for more similar input data [90]. This thesis considers distance functions only. Some distance functions even yield a metric space [31]; thus, metric index structures are applicable for improving the runtime of queries.

This section repeats both the concept of metric spaces, including simple examples, and the M-Tree, a well known metric index structure [35].

2.2.1 Metric spaces

A metric space consists of a set of objects and a function providing the distance between two objects [31, 47]. There are two main query types for searching in metric spaces: the ε-nearest neighbor query (also called *range queries*) and the k-nearest neighbor query. The *ε-nearest neighbor query* (ε-NN query) returns all elements from the dataset having a distance of at most ε to the query object. The *k-nearest neighbor query* (k-NN query) returns those k elements having the smallest distance to the query object.

Definition 1 (Pseudo metric space)**.** A *pseudo metric space* $(\mathbb{M}, d)$ consists of a set $\mathbb{M}$ and a distance function $d : \mathbb{M} \times \mathbb{M} \longrightarrow \mathbb{R}_{\geqslant 0}$ satisfying the following axioms:

$$\begin{aligned} \forall\ x, y \in \mathbb{M}\ &:\ d(x, y) = d(y, x). \\ \forall\ x, y, z \in \mathbb{M}\ &:\ d(x, z) \leqslant d(x, y) + d(y, z). \end{aligned}$$

Definition 2 (Metric space)**.** A *metric space* is a pseudo metric space which also satisfies the *reflexivity*:

$$\forall\ x, y \in \mathbb{M}\ :\ d(x, y) = 0 \iff x = y$$

Example 1. Note, if $\|\cdot\|$ is an arbitrary vector norm and $d(\cdot, \cdot)$ is defined as $d(u, v) := \|u - v\|$, then $(\mathbb{R}^k, d)$ is a metric space [47]. If $\|\cdot\|$ is an arbitrary matrix norm and $d(\cdot, \cdot)$ is defined as $d(M, M') := \|M - M'\|$ for all matrices $M, M' \in \mathbb{R}^{k \times k}$, then $(\mathbb{R}^{k \times k}, d)$ is a metric space.

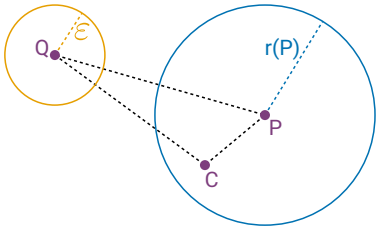


Figure 2.1: Sketch of three objects P, Q, C in a metric space where Q represents a query with radius ε and P represents a ball with center P and radius $r(P)$ which covers C.

The Euclidean vector distance is denoted by $\mathtt{d}_2(\cdot,\cdot) = \|\cdot - \cdot\|_2$, the Manhatten distance is denoted by $\mathtt{d}_1(\cdot,\cdot) = \|\cdot - \cdot\|_1$, and the maximum distance is denoted by $\mathtt{d}_\infty(\cdot,\cdot) = \|\cdot - \cdot\|_\infty$.

Example 2. Given a finite, undirected graph $\mathcal{G} = (V, E)$ with vertices V and edges $E \subseteq V \times V$, the shortest path distance $d : V \times V \to \mathbb{R}_{\geqslant 0}$ yields a metric space (V, d) [38].

Please note, that for each pseudo metric space $(\mathbb{M}, d)$ the equivalence relation $x \equiv y \iff d(x, y) = 0$ yields a metric space $(\mathbb{M}', d)$ where $\mathbb{M}'$ is the set of equivalence classes. Further metric distance functions on time series are discussed in Chapters 3 and 4.

2.2.2 Metric index structures

Similar to other index structures, the purpose of metric index structures is to improve the query runtime. Moreover, they are flexible and modular by indexing arbitrary data as long as it comes with a metric distance function. Despite common index structures as the B$^+$-Tree [36] for ordered data or the R*-Tree [16] for vector spaces, a metric index structure does not consider the properties of the elements themself but instead indexes the elements using the distances between them.

Metric index structures improve the runtime by pruning comparison of the query object to dataset objects whenever those can be excluded for sure. Therefore, they use the triangle inequality on the distances of the query object to already seen objects and the stored distances of objects within the dataset to estimate the distance of the query object to a candidate object in the dataset. See Figure 2.1 for an example of an ε-NN query where Q is the query element, P is an already seen object from the dataset, and C is a potential candidate for the result of the nearest neighbor query: The triangle inequality of distances $d(P, Q) \leqslant d(P, C) + d(C, Q)$ holds, i. e., $d(C, Q) \geqslant$

$d(P,Q) - d(P,C) \geqslant d(P,Q) - r(P) > \varepsilon$ and thus C is no candidate for the ε-NN query. The details of how metric spaces' properties are used exactly vary between different index structures, e. g., the M-Tree [35], the Cover Tree [19], the M-Index [76], and the VP-Tree [89]. Chávez et al. published a very detailed survey on searching and indexing techniques in metric spaces [31].

2.2.3 M-Tree

This section describes the basic structure and algorithms of the M-Tree [35], which is the basis for the SuperM-Tree in Chapter 5. Similar to the B^+-Tree, the R^*-Tree, and any other generalized search tree [46], the M-Tree is a balanced tree with nodes of a specific capacity.

Inner nodes contain a set of *routing objects*, each associated with one subtree. A routing object further describes a subset[1] of the metric space covering all objects in the corresponding subtree. Leaf nodes contain the actual entries (i. e., the metric objects that are to be indexed) and some optional user-defined data.

The insert algorithm first chooses the leaf node in which to insert the new element. If a node is overfilled after adding the new element, i. e., if it exceeds the capacity, then the node is split, and the old routing object in the parent node is replaced by two new routing objects covering the elements beneath the two new nodes. Thereby the size of the parent node increases by one. This process repeats recursively until either a node is not overfilled or the root node is split; in the latter, case a new root with two routing objects is created, and the height of the tree increases by one.

Similarly, the delete algorithm deletes objects from the leaf nodes and recursively applies the merge strategy on underfilled nodes.

Structure of M-Tree nodes

Leaf nodes store the indexed objects (*key objects*), whereas internal nodes store *routing objects*, which help in pruning and navigating through the tree.

Each routing object O_r is associated with a pointer to the root node $T(O_r)$ of its subtree, called the *covered tree* of O_r. The routing objects are also associated with a radius $r(O_r) \geqslant 0$ called the *covering radius* of O_r, as well as the distance $d(P(O_r), O_r)$ to their parent $P(O_r)$. All indexed objects in the covered tree of O_r are within the distance $r(O_r)$ from O_r.

Leaf nodes store the objects O_j and their distance $d(P(O_j), O_j)$ to the parent routing object. In real-world scenarios, leaf nodes contain additional information (e. g., data pointer or tuple identifier) per key object.

[1] The M-Tree describes a subset S of a metric space $(\mathcal{M}, d)$ with a center element p and a radius r: $S = \{x \in \mathcal{M} \mid d(x,p) \leqslant r\}$.

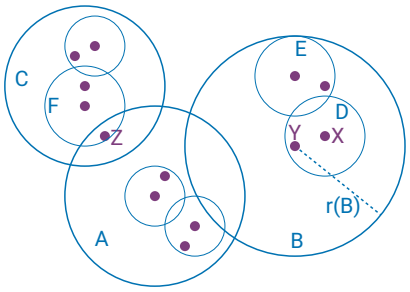


Figure 2.2: Sketch of the partitions within a metric space created by an M-Tree.

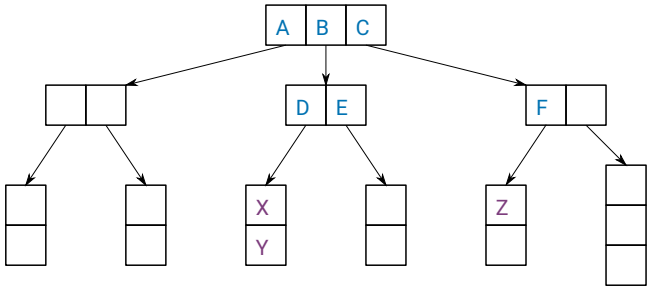


Figure 2.3: Sketch of the data structure tree created by the M-Tree that is sketched in Figure 2.2.

Figure 2.2 sketches an example for the structure of an M-Tree and its partitioning of the metric space. Note that the partitions may be overlapping. In the example, the root node contains three routing objects A, B, and C; the routing object B consists of the center object Y of the metric space and the covering radius $r(B)$; the covered subtree of B contains the routing objects D (with X as center object) and E; the subtree of D contains the elements X and Y. Note that both A and C cover the element Z. Sill, the insertion in the data structure takes place in one path only; in this case, it is inserted under the routing object F that is covered by C. Figure 2.3 sketches the data structure of this M-Tree.

Similarity queries

The M-Tree supports ε-nearest neighbor queries (range queries) and k-nearest neighbor queries. Since k-nearest neighbor queries adapt the search radius while traversing the tree, we only discuss ε-nearest neighbor queries here.

For an object $Q \in \mathcal{M}$, the range query `mtree.range`(Q, ε) selects all data base objects O_j with $d(O_j, Q) \leqslant \varepsilon$. Algorithm 1 provides the pseudo-code for the range query. It uses Lemma 1[2] and 2[3] to prune subtrees.

Lemma 1. If $d(O_r, Q) > r(Q) + r(O_r)$, then $d(O_j, Q) > r(Q)$ holds for each object O_j in the covered subtree of O_r.

Proof. Let O_j be an arbitrary but fixed object in the covered tree of O_r. By definition of metric spaces, the triangle inequality

$$d(O_r, Q) \leqslant d(O_r, O_j) + d(O_j, Q)$$

holds. The structure of the M-Tree requires $d(O_r, O_j) \leqslant r(O_r)$, thus

$$\begin{aligned} d(O_r, Q) &\leqslant r(O_r) + d(O_j, Q) \quad \text{and} \\ d(O_j, Q) &\geqslant d(O_r, Q) - r(O_r). \end{aligned}$$

Together with the prerequisite $d(O_r, Q) > r(Q) + r(O_r)$ of the lemma, the desired inequality

$$d(O_j, Q) > r(Q) + r(O_r) - r(O_r) = r(Q)$$

holds. □

Lemma 2. If $d(P(O_r), Q) > r(Q) + r(O_r) + d\big(P(O_r), O_r\big)$, then $d(O_j, Q) > r(Q)$ holds for each object O_j in the covered subtree of O_r.

Proof. Let O_j be an arbitrary but fixed object in the covered tree of O_r. Applying the triangle inequality twice yields

$$d\big(P(O_r), Q\big) \leqslant d\big(P(O_r), O_r\big) + d(O_r, O_j) + d(O_j, Q).$$

Since $d(O_r, O_j) \leqslant r(O_r)$ holds by definition of the structure of the M-Tree and $d(P(O_r), Q) > r(Q) + r(O_r) + d(P(O_r), O_r)$ is a prerequisite,

$$r(Q) + r(O_r) + d\big()P(O_r), O_r\big) < d\big(P(O_r), O_r\big) + r(O_r) + d(O_j, Q)$$

holds, and thus $r(Q) < d(O_j, Q)$. □

[2] In Euclidean vector spaces, Lemma 1 yields that no element of the covered subtree can be within the search range if the covered area of the subtree and the search area do not overlap.

[3] Lemma 2 is used as optimization in the implementation of the M-Tree: It might prune the subtree $T(O_r)$ without evaluating the distance of the query object Q to the routing object O_r. Therefore, it uses the distance value of the query object to the parent $P(O_r)$ of the routing object, which at that time is already evaluated (cf. Line 12 in Algorithm 1).

Algorithm 1 M-Tree: Range query

```
1  Algorithm: mtree.range
2  Input: node N (default: root node), query object Q,
       search radius r(Q)
3  R := ∅
4  if N is a leaf node
5    for each O_j ∈ N
6      if d(O_j, Q) ⩽ r(Q)
7        R := R ∪ {O_j}
8    return R
9  // else
10 for each O_r in N
11   if r(Q) < d(P(O_r), Q) − d(O_r, P(O_r)) − r(O_r)
12     skip // pruned using Lemma 2
13   if r(Q) < d(O_r, Q) − r(O_r)
14     skip // pruned using Lemma 1
15   R := R ∪ mtree.range(T(O_r), Q, r(Q))
16 return R
```

Building the M-Tree

The M-Tree is a generalized search tree (GiST [46]), i. e., algorithms for insertion and deletion of objects manage overflow and underflow of nodes [4] using split and merge operations.

The `mtree.insert` algorithm recursively descends the M-Tree down to the leaf node where to insert the object. For ambiguous cases, different strategies exist for choosing the routing object at each node. A simple strategy chooses the routing object with the closest distance to the new element. Further strategies are discussed in the original work [35]. At each routing object of the insertion path, the algorithm ensures that the covering radius $r(O_r)$ covers the newly inserted element by increasing the radius if necessary. Algorithm 2 provides the pseudo-code for the `mtree.insert` algorithm.

Split management Various split strategies exist for the M-Tree. They consist of a `promotion` and a `partition` algorithm (see Algorithm 3). Splitting a node N makes it two nodes N_1 and N_2, each getting a new parent routing object. The direct children of N are partitioned and distributed among the new nodes N_1 and N_2.

[4] An overflow or underflow occurs when a node's size exceeds the capacity or deceeds (i. e., is less than) a fixed fracture of the capacity, respectively

Algorithm 2 M-Tree: Insert

```
1  Algorithm: mtree.insert
2  Input: node N, object O
3  if N is a leaf node
4    N ≔ N ∪ {O}  // insert O to N
5  else
6    O_r ≔ arg min {d(O_r, O) | O_r ∈ N}
7    r(O_r) ≔ max {r(O_r), d(O_r, O)}
8    childSplit, O_1, O_2 ≔ mtree.insert(T(O_r), O)
9    if childSplit
10     N ≔ N \ {O_r} ∪ {O_1, O_2}
11 didSplit, O_1, O_2 ≔split(N)  // cf. Algorithm 3
12 if didSplit and N is root node
13   set new root node {O_1, O_2}
14 return didSplit, O_1, O_2
```

Algorithm 3 M-Tree: Split

```
1  Algorithm: mtree.split
2  Input: node N, object O
3  if |N| ⩽ capacity
4    return false, nil, nil
5  foundPromotion, O_1, O_2 ≔ promote(N)  // cf. Algorithm 4
6  if not foundPromotion:
7    return false, nil, nil
8  N_1, N_2 ≔ partition(N, O_1, O_2)  // cf. Algorithm 5
9  T(O_1) ≔ N_1
10 T(O_2) ≔ N_2
11 r(O_1) ≔ max {d(O_1, O) | O ∈ N_1}
12 r(O_2) ≔ max {d(O_2, O) | O ∈ N_2}
13 return true, O_1, O_2
```

First, the `promote` algorithm provides the two routing objects (one for each new partition) that both replace the old routing object in the parent node. The authors of the M-Tree figured that the choice of the promotion strategy is a trade-off between computation time for building the tree and computation time for querying the tree [35]. This thesis focuses on the strategy that promotes two objects minimizing the sum of the new nodes' covering radii (cf. Algorithm 4). If N is the root node, a new root node is created filled with the two promoted routing objects.

Given the promoted routing objects, the `partition` algorithm disjointly distributes the elements of the node N among the two nodes N_1 and N_2. Here, the *generalized hyperplane* strategy is presented, which puts each object to its nearest routing object (cf. Algorithm 5).

Algorithm 4 M-Tree: Promote

```
1  Algorithm: mtree.promote
2  Input: node N
3  e ≔ ∞
4  P1 ≔ nil
5  P2 ≔ nil
6  foundPromotion:=false
7  for each pair O1, O2 ∈ N
8    N1, N2 ≔ partition(N, O1, O2)  // cf. Algorithm 5
9    // get penalty for this partition
10   r1 ≔ max {d(O1, O) | O ∈ N1}
11   r2 ≔ max {d(O2, O) | O ∈ N2}
12   if r1 + r2 < e  // if new best candidate pair is found
13     foundPromotion:=true
14     P1 ≔ O1
15     P2 ≔ O2
16 return foundPromotion, P1, P2
```

Algorithm 5 M-Tree: Partition

```
1  Algorithm: mtree.partition
2  Input: node N, routing objects O1, O2
3  N1 ≔ {Oj ∈ N | d(O1, Oj) < d(O2, Oj)}
4  N2 ≔ N \ N1
5  return N1, N2
```

2.3 Data type: time series

Various definitions of the data type *time series* exist [32, 41]. Example distinctions include finite versus infinite time series and time series with time stamps versus simple sequences without time stamps. Furthermore, various compression techniques exist for representing time series [29, 41, 69]. This thesis only considers time series as finite sequences consisting of metric space elements and disregards any compression technique.

Examples with real-valued elements include stock market data and temperature measurements. Two-dimensional examples include trackings of GPS coordinates and gestures on touch screens. A sophisticated example that shows this data type's universality is tracking an element in a graph (e. g., a social web).

Definition 3 (Time series)**.** A time series T of length ℓ over a metric space $\mathbb{M}$ is a sequence $T = (t_1, \cdots, t_\ell)$ with $t_i \in \mathbb{M}$ for $1 \leqslant i \leqslant \ell$.

Note, that Definition 3 covers the default definition of real valued time series [32, 41]. The rest of this thesis considers $\mathbb{M} = \mathbb{R}^k$ for some $k \in \mathbb{N}$, although some results do not depend on this restriction.

The suffix of a time series defined by $\texttt{Tail}(T) \coloneqq (t_2, \cdots, t_n)$ removes the first element of the time series. A subsequence of T starting at index i with length k is denoted by $T_i^k \coloneqq (T_i, \cdots, T_{i+k-1})$.

2.4 Time series distance functions

While various distance functions on time series exist [33, 44, 65, 80, 82], this thesis only considers some of them and categorizes them by their semantics. The first type of distance function considered is *time-warping distance functions*. They aim at comparing time series while being robust against time distortion [41]. Various reasons cause time distortion, such as sensor inaccuracies, sensor failures, or even purpose by different people performing the same gesture with different accelerations on a touch screen. The second type of distance functions considered in this thesis is robust against isometric transformations, i. e., when a transformation translates, rotates, and reflects a time series without changing the shape of the time series [5]. This section provides examples for both of these types.

In the rest of this work, distance functions on time series are denoted via greek letters (e. g., $\delta : \mathbb{M}^{\mathbb{N}} \times \mathbb{M}^{\mathbb{N}} \longrightarrow \mathbb{R}_{\geqslant 0}$) and distance functions on their elements are denoted via arabic letters (e. g., $\mathtt{d} : \mathbb{M} \times \mathbb{M} \longrightarrow \mathbb{R}_{\geqslant 0}$). Given a distance function $\mathtt{d}$ on the elements of two time series S and T, the p-norm on the vector of all pairs $\mathtt{d}(s_i, t_i)$ is denoted via $\|\mathtt{d}\|_p \coloneqq \sqrt[p]{\sum_{i=1}^{n} \mathtt{d}(s_i, t_i)^p}$. If a particular distance function is meant, it is denoted via its name (e. g., `DK` or `DTW`).

2.4.1 Time-warping distance functions

Numerous time-warping distance functions considering time distortion (i. e., temporal displacements) exist. They all share the same approach for finding a good alignment between the elements of two time series. Common examples include Dynamic Time Warping (DTW) [80], the Levenshtein distance (a specific edit distance) [65], the Dog Keeper distance (DK) [40, 43], and the edit distance with real penalties (ERP) [33]. Although these functions are defined recursively (which yields exponential runtime in the length of the time series), algorithms using dynamic programming exist that reduce the complexity to quadratic runtime (e. g., [40, 80, 83]). This section repeats algorithms and properties of DTW, ERP, and DK. Although this thesis focuses on DTW and DK in later sections, ERP is included in this section to understand the common approach of time-warping distance functions.

The algorithms for the computation of DTW, ERP, and DK are very similar. They differ in how they handle a time-warping step and how they aggregate the distances of the elements of the time series: DTW and ERP sum up the values while the DK distance takes the maximum.

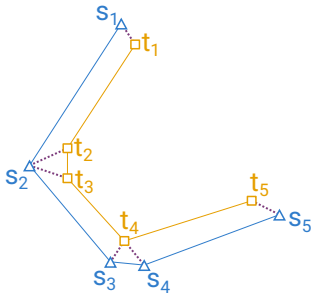
For a formal definition, let $S = (s_1, \cdots, s_m)$ and $T = (t_1, \cdots, t_n)$ be two time series, gap a globally constant element (e. g., 0 as proposed by the authors of ERP), and $\mathtt{d}(s, t)$ any distance function on the elements of the time series. The well known distance function DTW is defined as follows.

$$\begin{aligned}
\mathtt{DTW}(S, ()) &= \infty \\
\mathtt{DTW}((), T) &= \infty \\
\mathtt{DTW}((), ()) &= 0 \\
\mathtt{DTW}(S, T) &= \mathtt{d}(s_1, t_1) + \min \begin{cases} \mathtt{DTW}(\mathtt{Tail}(S), \mathtt{Tail}(T)) \\ \mathtt{DTW}(S, \mathtt{Tail}(T)) \\ \mathtt{DTW}(\mathtt{Tail}(S), T) \end{cases}
\end{aligned}$$

Following the path in the recursion tree which provides the minimum at each step provides a set of correspondences between the elements of S and T where the first correspondence is always (s_1, t_1). These correspondences define the *alignment* or *warping path* between S and T.

Example 3. Figure 2.4 sketches two time series S and T. The sequence $((s_1, t_1), (s_2, t_2), (s_2, t_3), (s_3, t_4), (s_4, t_4), (s_5, t_5))$ describes the alignment regarding DTW. Note that s_2 is associated with two elements of T, and t_4 is associated with two elements of S. Hence, the computation of DTW results in

$$\begin{aligned}
\mathtt{DTW}(S, T) &= \mathtt{d}(s_1, t_1) + \mathtt{d}(s_2, t_2) + \mathtt{d}(s_2, t_3) + \mathtt{d}(s_3, t_4) + \mathtt{d}(s_4, t_4) + \mathtt{d}(s_5, t_5) \\
&= 17.
\end{aligned}$$



t_5	18	17	12	9	3
t_4	17	10	3	4	13
t_3	13	2	7	10	18
t_2	9	3	12	14	19
t_1	2	11	18	19	19
	s_1	s_2	s_3	s_4	s_5

Figure 2.4: A sketched example showing the alignment (dotted lines) between two time series S and T regarding DTW (left) and the table of pairwise distances of the elements (right).

Note, that DTW corresponds to the Manhatten norm on the vector of distances regarding the best alignment:

$$\texttt{DTW}(S,T) = \left\| \Big(\texttt{d}(s_1,t_1), \texttt{d}(s_2,t_2), \texttt{d}(s_2,t_3), \texttt{d}(s_3,t_4), \texttt{d}(s_4,t_4), \texttt{d}(s_5,t_5) \Big) \right\|_1$$

Sometimes, the Euclidean norm is used instead [24, 64].

The ERP distance function differs from DTW by including gap elements to the time series on warping steps. Hence, the only element which may appear multiple times in the warping path is the gap element.

$$\begin{aligned}
\texttt{ERP}(S,()) &= \sum_{s_i \in S} \texttt{d}(s_i, \texttt{gap}) \\
\texttt{ERP}((),T) &= \sum_{t_i \in T} \texttt{d}(\texttt{gap}, t_i) \\
\texttt{ERP}((),()) &= 0 \\
\texttt{ERP}(S,T) &= \min \begin{cases} \texttt{d}(s_1,t_1) + \texttt{ERP}(\texttt{Tail}(S), \texttt{Tail}(T)) \\ \texttt{d}(\texttt{gap},t_1) + \texttt{ERP}(S, \texttt{Tail}(T)) \\ \texttt{d}(s_1,\texttt{gap}) + \texttt{ERP}(\texttt{Tail}(S), T) \end{cases}
\end{aligned}$$

The DK distance is similar to DTW and differs by taking the maximum distance instead of the sum along the warping path:

$$\begin{aligned}
\texttt{DK}(S,()) &= \infty \\
\texttt{DK}((),T) &= \infty \\
\texttt{DK}((s),(t)) &= \texttt{d}(s,t) \\
\texttt{DK}(S,T) &= \min \begin{cases} \max\{\texttt{d}(s_1,t_1), \texttt{DK}(\texttt{Tail}(S), \texttt{Tail}(T))\} \\ \max\{\texttt{d}(s_1,t_1), \texttt{DK}(S, \texttt{Tail}(T))\} \\ \max\{\texttt{d}(s_1,t_1), \texttt{DK}(\texttt{Tail}(S), T)\} \end{cases}
\end{aligned}$$

Hence, the alignments of the time series in Example 3 are the same for DK and DTW, but the distance value $\texttt{DK}(S,T) = 4$ differs from that of DTW. Note that this is not necessarily always the case.

It is easy to see that ERP and DK are symmetric. Moreover, they satisfy the triangle inequality [33, 43], which makes them pseudo metric distance functions. Still, ERP is not considered any further in this thesis since the distance value relies on the choice of the gap element, which results in unintuitive semantics. Furthermore, ERP is not truly invariant under time warping, as the following example shows.

Example 4. Let $S = (1, 2)$ and $T = (1, 1, 2)$ be two real-valued time series. The computation of $\texttt{ERP}(S,T)$ with $\texttt{gap} = 0$ results in the alignment $((s_1, t_1), (\texttt{gap}, t_2), (s_2, t_3))$ and thus $\texttt{ERP}(S,T) = 0 + 1 + 0 = 1$. At the same time, the expected result of a distance function that is truly invariant under time warping would be 0 since both time series follow the same path in space but have only temporal displacements (i. e., s_1 occurs twice in T).

Dog Keeper yields a metric: The DK distance is a special case of the Fréchet distance [43] that is defined on pairs of continuous trajectories $f : R \longrightarrow \mathbb{R}^n$ for some interval $R \subseteq \mathbb{R}$ and $n \in \mathbb{N}$. This section also abbreviates the Fréchet distance with DK. The proof that shows that the Fréchet distance satisfies the triangle inequality is written in French and an old style of mathematical language [43]. Hence, this section provides a new proof in a modern mathematical language that shows that the Fréchet distance (and thereby the Dog Keeper distance) satisfies the triangle inequality.

Let $\mathbb{M} \coloneqq \mathbb{R}^k$ be the space of states and $\mathsf{d} : \mathbb{M} \times \mathbb{M} \longrightarrow \mathbb{R}_{\geqslant 0}$ be a metric on all states. The set of all (piecewise continous) curves over $[0, 1] \subset \mathbb{R}$ is denoted by

$$\mathcal{T} \coloneqq \{f \colon [0, 1] \longrightarrow \mathbb{M}\}$$

and the set of all warping functions over $[0, 1]$ is denoted by

$$\Sigma \coloneqq \{\sigma \colon [0, 1] \longrightarrow [0, 1]\},$$

where all $\sigma \in \Sigma$ are continuous, strictly monotonically increasing, $\inf \sigma = 0$, and $\sup \sigma = 1$. For $f, g \in \mathcal{T}$, let $\delta_\infty(f, g) \coloneqq \max_{x \in [0,1]} \mathsf{d}(f(x), g(x))$ be the maximum distance of f and g.

Definition 4 (Fréchet Distance)**.** Let $f, g \in \mathcal{T}$ be two curves over $[0, 1]$. The Fréchet distance DK of f and g is defined as

$$\texttt{DK}(f, g) \coloneqq \inf_{\sigma,\tau \in \Sigma} \delta_\infty(f \circ \sigma, g \circ \tau)$$

Definition 4 is well defined since $\texttt{DK}(f, g)$ has 0 as lower bound and thus the infimum exists. It is easy to see that DK satisfies the symmetry. The following theorem claims that DK also satisfies the triangle inequality and therefore is a pseudo metric distance function.

Theorem 1. The Fréchet distance $\mathtt{DK}$ satisfies the triangle inequality, i. e.,

$$\forall f, g, h \in \mathcal{T} \colon \mathtt{DK}(f, h) \leqslant \mathtt{DK}(f, g) + \mathtt{DK}(g, h).$$

The following two lemmas are necessary to prove the triangle inequality. First, Lemma 3 shows that the δ_∞ distance does not change when applying the time warping on both curves. Then, Lemma 4 reduces the search to all warping functions applied to one time series only.

Lemma 3. Let $f, g \in \mathcal{T}$ be two arbitrary curves, and let $\sigma \in \Sigma$ be an arbitrary warping function. Then, the following equation holds:

$$\delta_\infty(f, g) = \delta_\infty(f \circ \sigma, g \circ \sigma)$$

Proof. Consider the mappping

$$\begin{aligned} \theta \colon [0, 1] &\longrightarrow \mathbb{R}_{\geqslant 0} \\ x &\longmapsto \mathsf{d}(f(x), g(x)). \end{aligned}$$

Then,

$$\begin{aligned} \delta_\infty(f, g) &= \sup\left(\theta([0, 1])\right), \text{ and} \\ \delta_\infty(f \circ \sigma, g \circ \sigma) &= \sup\left(\theta \circ \sigma([0, 1])\right) \end{aligned}$$

Since $\theta([0, 1]) = \theta(\sigma([0, 1]))$, the desired equation $\delta_\infty(f, g) = \delta_\infty(f \circ \sigma, g \circ \sigma)$ follows. □

Lemma 4. Let $f, g \in \mathcal{T}$ be two arbitrary curves. Then, the following equation holds:

$$\mathtt{DK}(f, g) = \inf_{\sigma \in \Sigma} \delta_\infty(f, g \circ \sigma)$$

Proof. Consider two sequences $(\sigma_i)_{i \in \mathbb{N}}$ and $(\tau_i)_{i \in \mathbb{N}}$ with $\sigma_i, \tau_i \in \Sigma$ for $i \in \mathbb{N}$, such that

$$\delta_\infty(f \circ \sigma_i, g \circ \tau_i) \xrightarrow{i \to \infty} \mathtt{DK}(f, g).$$

Since each σ_i is invertable, Lemma 3 can be applied on $\delta_\infty(f \circ \sigma_i, g \circ \tau_i)$ with σ_i^{-1}, i. e.

$$\begin{aligned} \delta_\infty(f, g \circ \tau_i \circ \sigma_i^{-1}) &= \delta_\infty(f \circ \sigma_i \circ \sigma_i^{-1}, g \circ \tau_i \circ \sigma_i^{-1}) \\ &= \delta_\infty(f \circ \sigma_i, g \circ \tau_i) \xrightarrow{i \to \infty} \mathtt{DK}(f, g). \end{aligned} \tag{2.1}$$

Equation 2.1 yields a sequence $(\theta_i)_{i \in \mathbb{N}} \coloneqq (\tau_i \circ \sigma_i^{-1})_{i \in \mathbb{N}}$ with $\theta_i \in \Sigma$ for $i \in \mathbb{N}$, such that $\delta_\infty(f, g \circ \theta_i) \xrightarrow{i \to \infty} \mathtt{DK}(f, g)$.

Furthermore,

$$\mathtt{DK}(f,g) = \inf_{\sigma,\tau\in\Sigma} \delta_\infty(f\circ\sigma, g\circ\tau) \leqslant \inf_{\theta\in\Sigma} \delta_\infty(f, g\circ\theta).$$

Hence, $\mathtt{DK}(f,g) = \inf_{\theta\in\Sigma} \delta_\infty(f, g\circ\theta)$. □

Proof of Theorem 1. Consider arbitrary but fixed curves $f, g, h \in \mathcal{T}$. Since $\mathtt{DK}(f,g) = \inf_{\sigma\in\Sigma} \delta(f, g\circ\sigma)$ holds by Lemma 4, an infinite sequence $(\sigma_i)_{i\in\mathbb{N}}$ exists with $\sigma_i \in \Sigma$ for all $i \in \mathbb{N}$, such that

$$\delta_\infty(f, g\circ\sigma_i) \xrightarrow{i\to\infty} \mathtt{DK}(f,g).$$

Analogously, a sequence $(\tau'_i)_{i\in\mathbb{N}}$ with $\tau'_i \in \Sigma$ for all $i \in \mathbb{N}$ exists, such that

$$\delta_\infty(g, h\circ\tau'_i) \xrightarrow{i\to\infty} \mathtt{DK}(g,h).$$

Considering the sequence $(\tau_i)_{i\in\mathbb{N}}$ with $\tau_i = \tau'_i \circ \sigma_i \in \Sigma$ and using Lemma 3 yields

$$\begin{aligned}\delta_\infty(g\circ\sigma_i, h\circ\tau_i) &= \delta_\infty(g\circ\sigma_i\circ\sigma_i^{-1}, h\circ\tau_i\circ\sigma_i^{-1}) \\ &= \delta_\infty(g, h\circ\tau'_i) \xrightarrow{i\to\infty} \mathtt{DK}(g,h).\end{aligned}$$

Recall, that $(\mathcal{T}, \delta_\infty)$ is a metric space, thus the triangle inequality holds for each $i \in \mathbb{N}$:

$$\delta_\infty(f, h\circ\tau_i) \leqslant \delta_\infty(f, g\circ\sigma_i) + \delta_\infty(g\circ\sigma_i, h\circ\tau_i)$$

Since $\mathtt{DK}(f,h) = \inf_{\tau\in\Sigma} \delta_\infty(f, h\circ\tau)$, the triangle inequality holds:

$$\begin{aligned}\mathtt{DK}(f,h) &\leqslant \lim_{i\to\infty} \delta_\infty(f, h\circ\tau_i) \\ &\leqslant \lim_{i\to\infty} \delta_\infty(f, g\circ\sigma_i) + \lim_{i\to\infty} \delta_\infty(g\circ\sigma_i, h\circ\tau_i) \\ &= \mathtt{DK}(f,g) + \mathtt{DK}(g,h)\end{aligned}$$

□

2.4.2 Congruence of time series

This section first outlines geometric point pattern matching to induce an understanding of the latter congruence of time series. The introduction to the congruence of time series then uses the same notation.

Two sets A and B are said to be *congruent* iff $\mu(A) = B$ holds for some *isometry* μ (i. e., for some distance preserving function) [27]. Isometric functions consist of rotation, translation, reflection, and any concatenation of these operations. Note that the correspondence of the points of A to the points of B is not known beforehand.

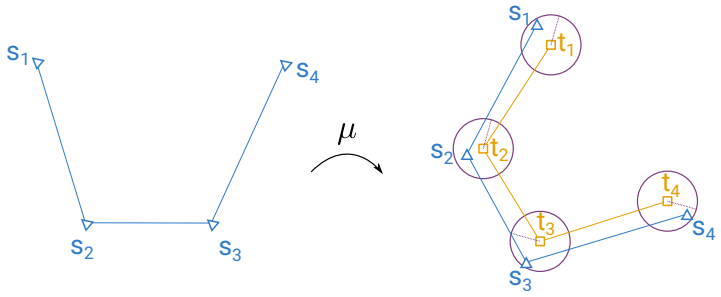


Figure 2.5: A time series T (right) with ε-balls around each element and a time series S (left) which is rotated and translated via transformation μ to fit into the ε-balls of T.

Various problem statements regarding congruence of sets exist [4, 5]. They distinguish between exact and approximated congruence. More relevant to this thesis is the approximated case which is more realistic by allowing errors: The points of A are to be mapped into ε-neighborhoods of the points of B, i. e., A and B are *ε-approximately congruent* iff each $\mu(a)$ ($a \in A$) is in the ε-neighborhood of a distinct $b \in B$ for some isometry μ (cf. Figure 2.5).[5]

Given sets A and B, the CONGRUENCE problem is to decide whether A and B are congruent [2, 7, 27]. Given a further error value ε, the approximated CONGRUENCE problem with tolerance ε is to decide whether A and B are ε-approximately congruent [4–6, 50]. This thesis aims at finding a minimal ε (and possibly the corresponding transformation function μ), such that A and B are (approximately) congruent with tolerance ε [4,5].[6] As stated in the two referenced surveys [4,5], only a little research exists in finding a minimal tolerance ε for the approximated congruence. Furthermore, most of the research either considers two- and three-dimensional point sets or considers the exact CONGRUENCE problem in high dimensional space.

This section formally introduces the congruence of time series over Euclidean vector spaces. Thereby, two time series S and T are said to be *congruent*, iff $\mu(S) = T$ (i. e., $\mu(s_i) = t_i$ for each $1 \leqslant i \leqslant \#S$) holds for some isometry μ. The difference to point pattern matching is that the correspondence between the points of the two time series is already known. In the approximated case, an error of ε is tolerated, i. e., S and T are *congruent with tolerance* ε iff $\|s_i - \mu(t_i)\| \leqslant \varepsilon$ holds for some isometry μ and each $1 \leqslant i \leqslant \#S$. In general, taking the maximum of some error values is vulnerable to noise and can be improved by summing the error values. Therefore, it is useful to generalize the congruence: S and T are congruent

[5] Most commonly, congruence problems are considered in Euclidean vector spaces.
[6] This thesis aims at finding minimal distances regarding congruence of time series.

with tolerance ε regarding some norm $\|\cdot\|$, iff

$$\inf_{\mu} \{\|\mathtt{d}_2\|(S, \mu(T))\} \leqslant \varepsilon$$

where $\|\mathtt{d}_2\|(S, \mu(T)) \coloneqq \|(\mathtt{d}_2(s_i, \mu(t_i)))_{1 \leqslant i \leqslant \#S)}\|$, i. e., the Euclidean distance $\mathtt{d}_2$ is applied on the pairs of corresponding elements while the norm $\|\cdot\|$ is applied on the vector of these distances. The time complexity (regarding the dimensionality as the parameter) of any algorithm computing the congruence regarding the sum norm is analyzed in Chapter 4. Chapter 4 further proposes fast algorithms computing lower bounds for the exact value.

2.5 Time Series Datasets and Synthesizers

Fast algorithms for time series distance functions are necessary in order to achieve fast query runtimes. There are some evaluations of different time warping distance functions on different datasets [85], but their performance has not been evaluated concerning growing dimensionality. Furthermore, datasets are necessary for the evaluation of the proposed congruence distance functions (see Chapter 4) as well as the proposed index structure (see Chapter 5).

Computation time of distance function: To evaluate the computation time of a distance function concerning growing dimensionality, datasets with similar properties (e. g., size of the dataset, length of time series, data distribution) but different dimensionality are necessary. Existing datasets of different dimensionality exist but have different properties. This thesis uses synthesized data to achieve similar properties on datasets with different dimensionality.

Tightness of lower bounds: Nearest neighbor queries are also accelerated by pruning distance computations using cheap lower bounds [58, 78]: If a lower bound claims a large distance, there is no need to compute the exact but expensive distance value. The lower bound proposed by Keogh was extended to multi-dimensional time series [71] (here called $\mathtt{LB}_{\mathtt{Box}}$), but there is no evaluation available regarding growing dimensionality. Again, having datasets with similar properties for different dimensionality is necessary for these evaluations. Those datasets could also be used to evaluate pruning strategies of index structures specialized to `DTW` [74] or metric index structures [19, 23, 35, 76] applicable for metric time series distances (e. g., `DK` [9] and `ERP` [33]).

Accuracy of classifiers: Nearest neighbor search appears in supervised machine learning, for example, one-nearest neighbor classifiers. Labeled datasets are necessary to compare the properties of two classifiers using two different distance functions. Since the comparison is meaningless when both classifiers are perfectly accurate, the difficulty of classification tasks needs to be controllable. Thus, when designing dataset generators to compare the strength of two classifiers, parameters to control the difficulty (e. g., the distortion or fuzziness of data) need to be implemented.

Time distortion of trajectories: Some of the evaluation tasks addressed in this thesis consider time-warping distance functions that require time distorting dataset generators. Otherwise, evaluations would prefer implementations of distance functions that have an advantage in comparing perfectly aligned time series. The results of those evaluations were not transferable to datasets with time distorted time series.

Contribution: This thesis proposes two dataset generators for multi-dimensional, labeled, and time distorted time series to make such evaluation scenarios feasible. Both generators provide tuning parameters to control the difficulty of generated classification tasks.

The rest of this section is structured as follows. Section 2.5.1 presents the first dataset generator which is an extension of the well known cylinder-bell-funnel dataset [79]. Section 2.5.2 presents the `RAM` dataset generator which adapts ideas from Brownian motions. The datasets are evaluated using `DTW` in Section 2.5.4 in order to confirm that `DTW` yields an applicable one-nearest neighbor classifier and that its classification errors correlate with the difficulty parameters.

2.5.1 Cylinder-Bell-Funnel

N. Saito proposed the well known one-dimensional cylinder-bell-funnel dataset in his Ph.D. thesis [79]. It is an artificial dataset consisting of three different time series classes: cylinder, bell, and funnel.

For the time series synthesizer, let $\ell > 0$ be a fixed length of the time series, and $\mathcal{N}$ be a standard normal distributed random variable. Furthermore, fixate a and b uniformly distributed over $\left[\ell \cdot \frac{1}{8}, \ell \cdot \frac{2}{8}\right]$ and $\left[\ell \cdot \frac{6}{8}, \ell \cdot \frac{7}{8}\right]$, respectively, and $\nu = 6 + \mathcal{N}$, each of which is sampled per generated time series. Each time series has a prefix $\mathcal{P}$ of length a and suffix $\mathcal{S}$ of length $\ell - b$ containing standard normal distributed random numbers $(\mathcal{N}, \cdots, \mathcal{N})$.

The middle parts of random *cylinder* ($\mathcal{C}$), *bell* ($\mathcal{B}$), and *funnel* ($\mathcal{F}$) time series are a plateau, a rising linear function, and a falling linear function,

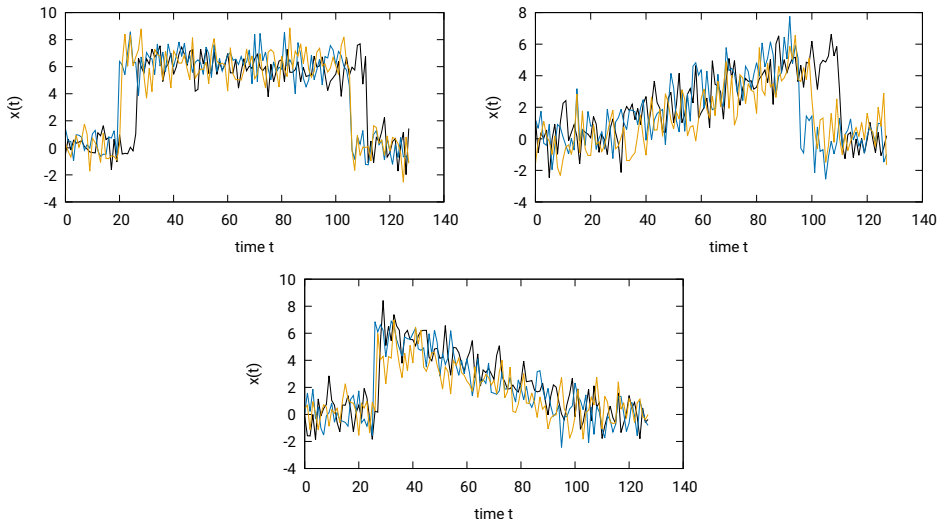


Figure 2.6: Three examples for cylinder (left top), bell (right top), and funnel (bottom) time series, respectively.

respectively. Their length is $b - a$; thus,

$$\mathcal{C} \coloneqq \mathcal{P} \times (\cdots, \nu + \mathcal{N}, \cdots) \times \mathcal{S}$$
$$\mathcal{B} \coloneqq \mathcal{P} \times \left(\cdots, \nu \cdot \frac{i-a}{b-a} + \mathcal{N}, \cdots\right) \times \mathcal{S}$$
$$\mathcal{F} \coloneqq \mathcal{P} \times \left(\cdots, \nu \cdot \frac{b-i}{b-a} + \mathcal{N}, \cdots\right) \times \mathcal{S}$$

with $0 \leqslant i \leqslant b - a$, where $\nu = 6 + \mathcal{N}$ is chosen once per time series (cf. Figure 2.6).

In this thesis, the `CBF` dataset generator is canonically extended to generate multi-dimensional time series by generating one of these types for each dimension with the same starting and ending positions a and b (cf. Figure 2.7 for a two-dimensional example). Thus, the input parameters are the length of the time series and a vector $\mathbf{t}$ with values c (for cylinder), b (for bell), and f (for funnel) that labels the type to synthesize per dimension. Given the dimensionality n, the multi-dimensional `CBF` generator produces a maximum number of 3^n different classes, which is the set of all combinations of c, b, and f.

2.5.2 Random Accelerated Motion Generator

The `RAM` generator produces classes by first generating *base time series* using impulse-driven motions with random acceleration. It then generates *Representatives* of the classes by distorting the base time series in space and time.

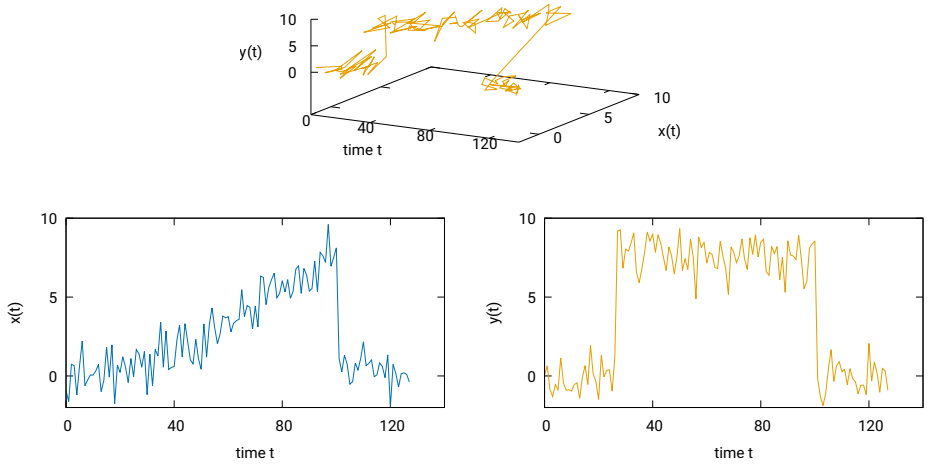


Figure 2.7: An example for a two-dimensional CBF time series (top). Its first dimension is a bell (left bottom) and its second dimension is a cylinder (right bottom).

Base Time Series Generator

The Brownian motion is a standard model of random motion in physics, e. g., to model molecules' movement in gases. It delivers each next position of a molecule as a randomized position around the current position.

To achieve more *curve-like* time series, an impulse vector is kept and added to the current position to obtain the next position. In each step, a normal distributed random vector adds to the impulse vector. Hence the time series are generated by distorting the first derivative instead of the current position.

To model edges in the generated time series, the movement is restricted to a ball with a constant radius R. When the time series is about to leave the restricted area, it instead bounces off the sphere; thus, the time series remains interior. Algorithm 6 provides the pseudocode for data generation and Figure 2.8 shows an example for a two-dimensional time series. The algorithm uses `uniformBall` to obtain a uniformly distributed vector from the interior of a unit sphere and `uniformSphere` to obtain a uniformly distributed vector on a union sphere.

Generating Representatives

Representatives of a class corresponding to a base time series are generated by distorting the time series in space and time separately.

Algorithm 6 Random Accelerated Motion Generator

```
1  Algorithm: rambase
2  Input: length l, dimensionality n, radius r
3  Output: time series S
4
5  v ≔ 0 ∈ ℝ^n
6  ν ≔ normal(0, 1)
7  s_1 ≔ uniformBall(r)
8  for i from 2 to l
9      v ≔ v + uniformSphere(r)
10     s_i ≔ s_{i−1} + v
11     if ‖s_i‖_2 > r
12         // rescale s_i to stay within the ball with radius r
13         s_i ≔ s_i · r / ‖s_i‖_2
14         v ≔ reflect v on sphere at point s_i
15 return S ≔ (s_1, ··· , s_l)
```

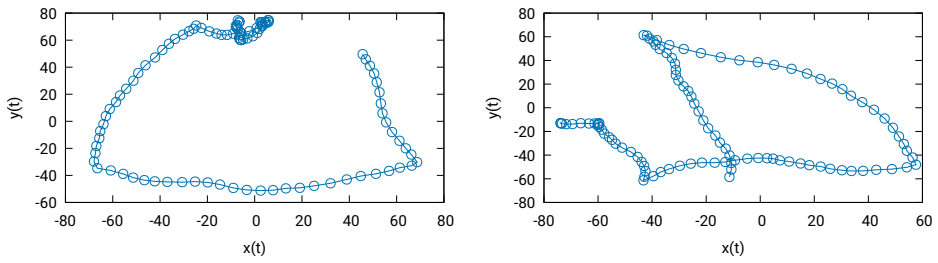


Figure 2.8: Two examples for a random accelerated motion in a two-dimensional vector space.

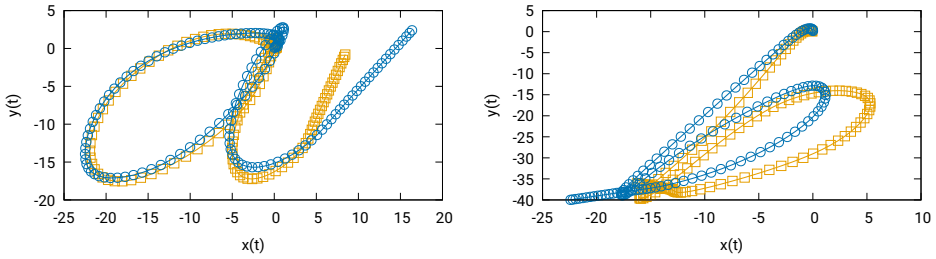


Figure 2.9: Two time series representatives from two classes of the character trajectories dataset, respectively [67].

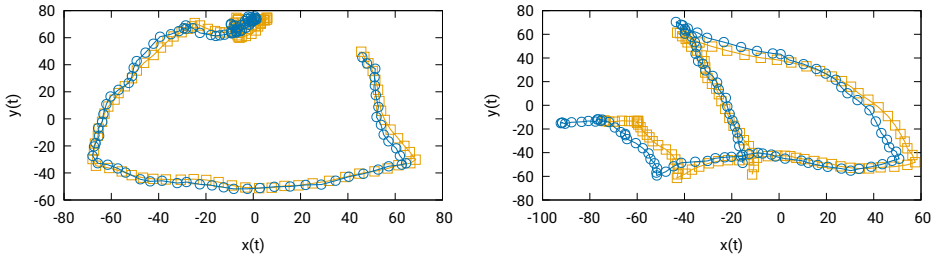


Figure 2.10: Two time series (yellow) and their spatio-distorted derivative (blue); length: 100; dimensionality: 2; distortion: 5 (left) and 25 (right).

Distortion in space Figure 2.9 shows examples from the character trajectories dataset [67]. Naturally, the representatives of a class do not match exactly. The `RAM` generator imitates this property by adding normally distributed noise to the first derivative of the time series.[7] The generator also limits the new time series' maximum distance to the base time series by a distortion parameter to prevent large divergence on long time series. Figure 2.10 shows two example time series and a copy for each with distorted derivatives.

Distortion in time In order to apply distortion in time, the time series is interpreted as a continuous curve. Points between two adjacent points of the time series are computed using linear interpolation. Then, the curve is reparameterized in terms of the arc length instead of the time. Finally, the time distorted time series consists of the first element of the time series, a set of points uniformly distributed on the reparameterized curve, and the last element of the time series. Algorithm 7 provides the pseudocode for the time distortion and Figure 2.11 shows two examples of time distorted time series.

[7] Also, the first element is distorted to prevent time series of the same class from equaling in the first position.

Algorithm 7 Time Distortion of a Time Series

```
Algorithm: timedistortion
Input: time series s of length L
Output: time series s̃

// get arc length up to each point of the time series
ℓ_i = Σ_{j=1}^{i} ‖s_j − s_{j−1}‖_2
// get uniformly distributed values along the complete arc
t = (0, ℓ_{L−1})
repeat L − 2 times
    t = t ∘ uniform([0, ℓ_{L−1}])
// interpolate between reparameterized points
s̃ = ()
for x in sort(t)
    // find correct index
    i = min {i | ℓ_i ⩽ x < ℓ_{i+1}}
    // interpolation parameter
    u = (x − ℓ_i) / (ℓ_{i+1} − ℓ_i)
    s̃ = s̃ ∘ ((1 − u) · s_i + u · s_{i+1}) < ++ >
return s̃
```

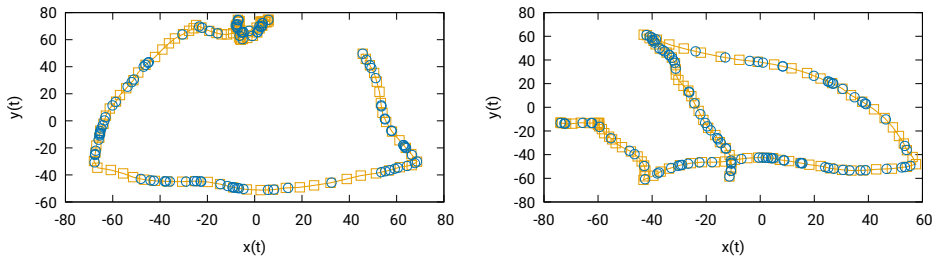


Figure 2.11: Two time series (yellow squares) and their time distorted versions (blue circles); length: 100; dimensionality: 2.

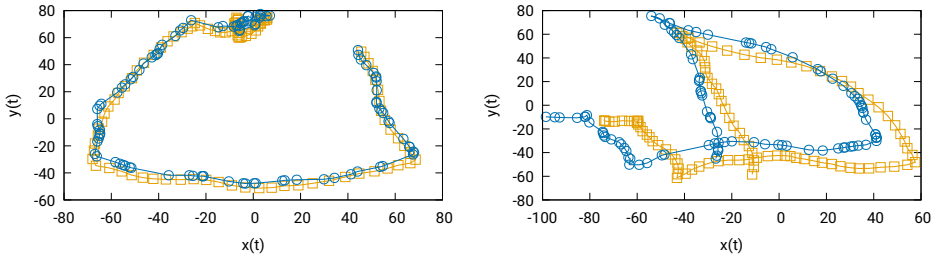


Figure 2.12: Two base time series (yellow squares) and their distorted versions (blue circles); length: 100; dimensionality: 2; radius: 75, distortion: 5 (left) and 25 (right).

2.5.3 Dataset generator

As mentioned at the beginning of this section, the base time series generator produces the different classes. The time distortion and space distortion algorithms generate the actual representatives of the classes.

Hence, to generate a dataset $\mathcal{D}$ with C classes, each of which has N representatives, the generator first calls `rambase` C times to generate $T_{i,0}$ $(1 \leqslant i \leqslant C)$. Then, for each $1 \leqslant i \leqslant C$, the generator calls `timedistortion` and `spacedistortion` N times to generate $T_{i,j}$ $(1 \leqslant j \leqslant N)$. The dataset consists of each $T_{i,j}$ for $1 \leqslant i \leqslant C$ and $1 \leqslant j \leqslant N$.

$$\begin{aligned}
\mathcal{C} &\coloneqq \{T_{i,0} = \texttt{rambase}(L, n, r) \mid 1 \leqslant i \leqslant C\} \\
\mathcal{R}_i &\coloneqq \big\{T_{i,j} = \texttt{spacedistortion}\big(\texttt{timedistortion}(T_{i,0}), D\big)\big\} \\
\mathcal{D} &\coloneqq \bigcup_{i=1,\cdots,j} \mathcal{R}_i
\end{aligned}$$

where n is the desired dimensionality, r is the radius of the bounding sphere, L is the desired length of the time series, and D is the desired degree of distortion within each class. Figure 2.12 shows two examples of a base time series and their time and space distorted representatives.

2.5.4 Evaluation

The two dataset generators `CBF` and `RAM` are supposed to produce datasets that are applicable for benchmarking time-warping distance functions. In order to verify whether the datasets are suitable for this task, two one-nearest neighbor classifiers are applied: a classifier based on the Euclidean distance function (`ED`) and a `DTW`-based classifier. If the scores of the `ED`-based classifier are low while the `DTW`-based classifier achieves high scores, it is reasonable to assume that the datasets are suitable for benchmarking time-warping distance functions. Furthermore, this section's evaluation verifies whether the parameters controlling the classification tasks' difficulty correlate with the `DTW`-based classifier's error.

Cylinder-Bell-Funnel

The `CBF` dataset has no distortion parameter, which makes it harder to evaluate the classification strength of time-warping distance functions. However, the classification task's difficulty can be influenced by changing the number of representatives per class.

The heatmap in Figure 2.13 shows that the classification score increases with growing class size for each dimensionality of the `CBF` dataset. However, Figure 2.14 shows that there are a few cases where the classification score slightly decreases with growing class size. Hence, the class size can roughly control the classification task's difficulty, but it may not be considered a reliable parameter for this purpose.

Figure 2.13 also shows that the classification strength decreases on higher dimensionality. Various experiments showed that this behavior neither depends on the length of the time series nor the number of classes.

Figure 2.13 and 2.14 both show that `DTW` achieves remarkably higher classification scores than the Euclidean distance. Hence, it appears that time-warping distance functions (such as `DTW`) are necessary to achieve good classification scores on this dataset.

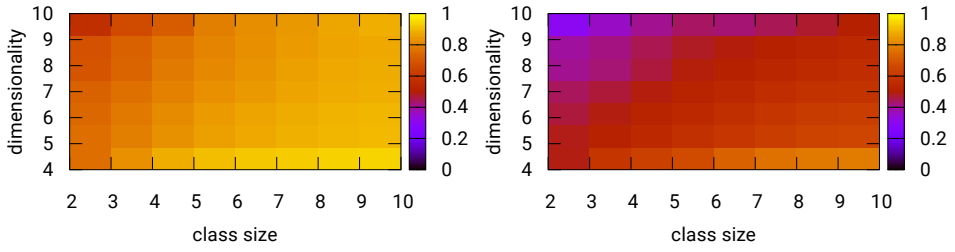


Figure 2.13: Classification scores for `CBF` dataset using `DTW` (left) and `ED` (right); length: 125; number of classes: 27

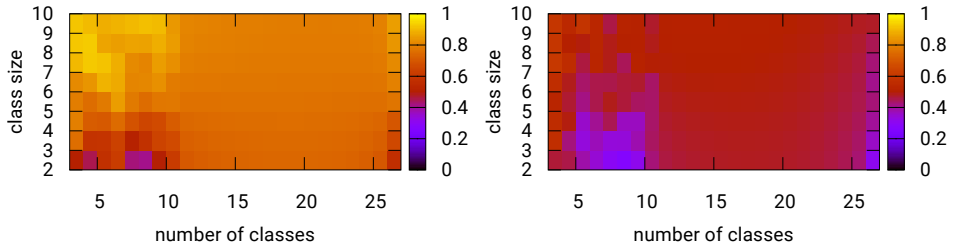


Figure 2.14: Classification score for `CBF` dataset using `DTW` (left) and `ED` (right); length: 125; dimensionality: 10

Random Acceleration Motion

As mentioned in Section 2.5.2, RAM generates datasets of multi-dimensional labeled time series. Similar to the CBF datasets, Figure 2.15 shows that the classification score increases with growing class size. Also, this generator has a distortion parameter to control the noisiness of the time series. This parameter of the RAM synthesizer impacts the classification score as expected: The score decreases with increasing distortion.

Regarding the dimensionality, the RAM synthesizer seems to be complementary to the CBF synthesizer, as the classification scores increase with growing dimensionality (c. f. Figure 2.16).

Figure 2.15 furthermore shows that the DTW distance performs better than the Euclidean distance. Again, it appears that time-warping distance functions are better suited for solving classification tasks on these datasets.

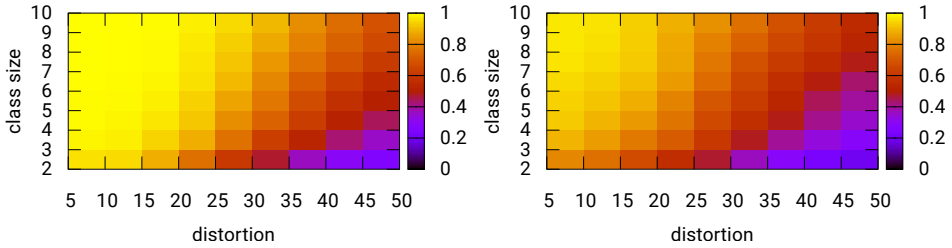


Figure 2.15: Classification score using DTW (left) and ED (right) for an example parameter set: radius 50, length 100, dimensionality: 3, number of classes 200.

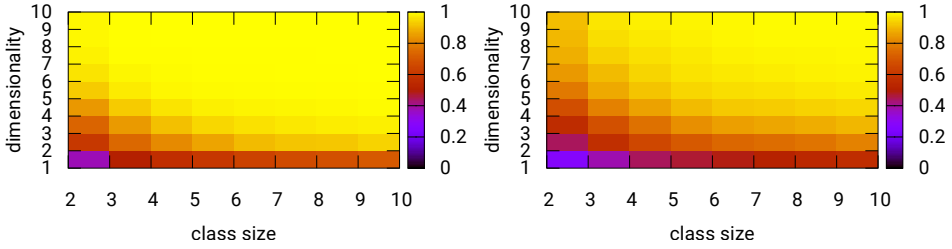


Figure 2.16: Classification score using DTW (left) and Euclidean distance (right) for an example parameter set: radius 50, length 100, number of classes 200, distortion: 5.

2.5.5 Conclusion

This section introduced two new dataset generators producing multi-dimensional labeled time series. The datasets are applicable for classification tasks using time-warping distance functions such as Dynamic Time Warping (`DTW`). Both generators provide parameters adjusting the difficulty of the classification task. Since the classification scores using `DTW` increase with growing dimensionality on the `RAM` datasets while decreasing on the `CBF` dataset, they seem to have some complementary properties. Thus, both synthesizers seem to be well suited for evaluating classifiers using time-warping distance functions concerning the dimensionality. Still, further evaluations using the `CBF` generator need to consider the unreliability of its difficulty parameter.

Chapter 3

Time-Warping Distance Functions

Dynamic Time Warping (DTW) is one of the most used distance functions to compare time series, e. g., in nearest neighbor classifiers [41]. However, fast state-of-the-art algorithms only compare one-dimensional time series efficiently. One of these state-of-the-art algorithms uses a lower bound ($\mathtt{LB}_{\mathtt{Keogh}}$) introduced by E. Keogh to prune expensive DTW computations [78]. This section presents $\mathtt{LB}_{\mathtt{Box}}$ as a canonical extension to $\mathtt{LB}_{\mathtt{Keogh}}$ on multi-dimensional time series. A conceptual and experimental evaluation of the performance of $\mathtt{LB}_{\mathtt{Box}}$ shows that its pruning power decreases drastically with increasing dimensionality; thus, an alternative to DTW with $\mathtt{LB}_{\mathtt{Box}}$ is necessary for multi-dimensional time series. Therefore, this section also proposes a new algorithm for the Dog Keeper distance (DK), an alternative distance function to DTW that outperforms DTW with $\mathtt{LB}_{\mathtt{Box}}$ by more than one order of magnitude on high-dimensional time series.

3.1 Introduction

Multimedia retrieval is a typical application that requires finding similar time series to a given input time series. Motivating examples are gesture recognition with modern virtual reality motion controllers, GPS tracking, speech recognition, and classification of handwritten letters.

All these examples share the property that the time series of the same classes (e. g., same written characters, or same gestures) follow the same path in space but have some temporal displacements. Tracking the GPS coordinates of two cars driving the same route from A to B is an illustrative example. These two time series shall be recognized as similar, although driving style, traffic lights, and traffic jams might result in temporal differences. *Time-warping distance functions*, such as Dynamic Time Warping (DTW) [80], the Dog Keeper distance (DK) [40], and the edit distance with real

penalties (ERP) [33], respect this semantic requirement. They map pairs of time series representing similar trajectories to small distances.

Unfortunately, these time-warping distance functions usually have quadratic runtime [15, 25, 26]. A common approach for improving the runtime of nearest neighbor queries is pruning as much distance computations as possible using lower bounds to the expensive distance function: If the lower bound exceeds a certain threshold (e. g., the largest distance to the k-nearest neighbors found so far), it implies a larger value for the expensive distance function, and thus the expensive computation can be skipped.

E. Keogh proposed one of the state-of-the-art algorithms for nearest neighbor queries with DTW on one-dimensional time series. His main contribution is the lower bound function $\mathtt{LB}_{\mathtt{Keogh}}$ [58,78] successfully pruning many expensive DTW computations under certain limitations.

Contributions: The contributions of this chapter are the following:

- This chapter introduces $\mathtt{LB}_{\mathtt{Box}}$, a canonical extension to $\mathtt{LB}_{\mathtt{Keogh}}$ for multi-dimensional time series in Section 3.3, i. e., $\mathtt{LB}_{\mathtt{Box}}$ is a lower bound for DTW on multi-dimensional time series and equals $\mathtt{LB}_{\mathtt{Keogh}}$ on one-dimensional time series.[1]

- Section 3.3 examines $\mathtt{LB}_{\mathtt{Box}}$ theoretically and shows that it suffers from an effect similar to the curse of dimensionality. Section 3.5 confirms the theoretical results with experiments on two synthesized and several real-world datasets.

- Section 3.4 proposes a new algorithm to compute the DK distance that is faster than DTW with $\mathtt{LB}_{\mathtt{Box}}$ by more than one order of magnitude on high-dimensional time series. The comparisons of DK and DTW with $\mathtt{LB}_{\mathtt{Box}}$ in terms of accuracy and computation time are presented in Section 3.5.

3.2 Preliminaries

DTW and DK are distance functions on time series [80] (cf. Section 2.4.1). Their benefit is a dynamic time alignment; thus, they are robust against time distortion or temporal displacements.

Well known algorithms computing DTW and DK in quadratic time exist [40, 80]. Although the algorithms are defined on real-valued time series, they can be extended canonically to time series over the metric space $(\mathbb{R}^k, \mathtt{d})$. These algorithms first build the cross product of both time series S and T using the distance function d. The resulting *distance matrix* consists of entries $\mathtt{d}(s_i, t_j)$, where $\mathtt{d}(s_1, t_1)$ is in the bottom left cell and $\mathtt{d}(s_m, t_n)$ is in the

[1] Remark, that this approach already exists as a technical report [71].

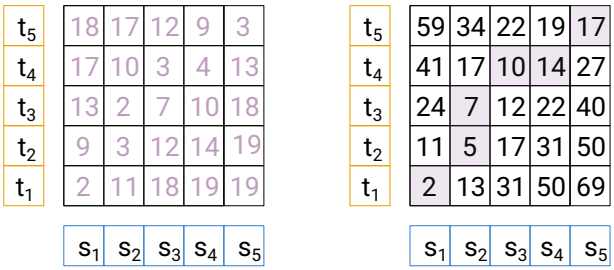


Figure 3.1: A table of pairwise distances of the two time series in Example 3 (left) and the corresponding warping matrix after computation of DTW (right). The highlighted cells in the warping matrix represent the warping path (i. e., the alignment of both time series).

top right cell (cf. Figure 2.4 in Example 3). The algorithms then compute the cheapest path from the bottom left to the top right cell by cell. For each cell, they replace its value with a combination of that value and the smallest value of one of the possible predecessors: the left, the lower, and the left lower neighbor cell, yielding the *warping matrix*. Figure 3.1 shows the warping matrix for the time series in Example 3. Unfortunately, any algorithm computing DTW or DK has quadratic runtime complexity in worst case [25, 26].

3.2.1 DTW and the Sakoe-Chiba band

The Sakoe-Chiba band changes the semantics of DTW to $\texttt{DTW}_r$ by constraining the possible paths in the warping matrix to a diagonal band of a particular bandwidth r [80]. Thus, time-warping is constrained, but the computation time decreases since a considerable part of the distance matrix does not need to be considered. In other words, the computation of DTW_r with a Sakoe-Chiba band of bandwidth r only considers the diagonal and r upper and lower co-diagonals. Figure 3.2 shows the computation of DTW and $\texttt{DTW}_r$ for the time series in Example 3 with and without a Sakoe-Chiba band, respectively. For a real value $0 < R < 1$, $\texttt{DTW}_R$ derives the bandwidth as a fracture R of the length of the time series. In his work, E. Keogh proposed a bandwidth of $R = 10\%$ [58]. In the example of Figure 3.2, the best warping path also fits in the Sakoe-Chiba band, and thus the result remains the same. Although the runtime complexity remains quadratic, smaller bandwidths quadratically decrease the runtime, e. g., for a bandwidth of $R = 10\%$, the Sakoe-Chiba band only covers $10\% * 10\% = 1\%$ of the warping matrix, and thus only 1% of the cells need to be computed. On the other hand, the distance function gets less flexible regarding time warping.

t_5	59	34	22	19	17
t_4	41	17	10	14	27
t_3	24	7	12	22	40
t_2	11	5	17	31	50
t_1	2	13	31	50	69
	s_1	s_2	s_3	s_4	s_5

t_5				19	17
t_4			10	14	27
t_3		7	12	22	
t_2	11	5	17		
t_1	2	13			
	s_1	s_2	s_3	s_4	s_5

Figure 3.2: The warping matrix after computation of plain `DTW` (left) and $\texttt{DTW}_r$ with Sakoe-Chiba band and a bandwidth of $r = 1$ (right). The highlighted cells in the warping matrix represent the warping path (i. e., the alignment of both time series).

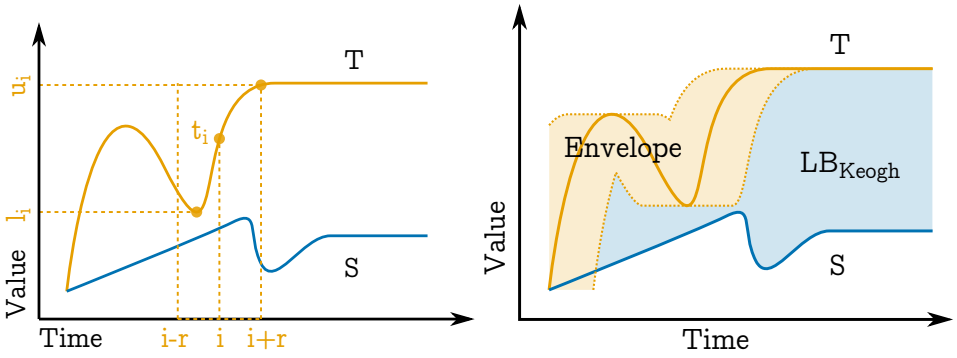


Figure 3.3: The envelope functions (computation sketched on the left) are used to compute the lower bound $\texttt{LB}_{\texttt{Keogh}}$ (right).

3.2.2 Keogh's lower bound for `DTW`

Cheap lower bounds are used to prune many complete `DTW` computations and thus improve the overall computation time of nearest neighbor queries: If the lower bound already exceeds a desired threshold, then the final result of the expensive `DTW` distance function will be larger as well. Keogh proposed one of the most successfull lower bounding functions *to the constrained* $\texttt{DTW}_r$ [58, 78]. His lower bound $\texttt{LB}_{\texttt{Keogh}}$ depends on the Sakoe-Chiba Band that constrains finding the best time alignments to a maximum time distance of a specific bandwidth.

The derivation of this lower bound is the following (cf. Figure 3.3 for an illustration): Consider two time series $S = (s_1, \cdots, s_m)$ and $T = (t_1, \cdots, t_n)$ and map each $s_i \in S$ to the interval of all possible aligned values within the

time range (i. e., the Sakoe-Chiba band) $i - r$ and $i + r$:

$$\begin{aligned} l_i &\coloneqq \min\{t_{i-r}, \cdots, t_{i+r}\} \\ u_i &\coloneqq \max\{t_{i-r}, \cdots, t_{i+r}\} \end{aligned} \tag{3.1}$$

Summing up the shortest square distances of s_i to each interval $[l_i, u_i]$ is a lower bound for $\mathtt{DTW}_r$[2] since $\mathtt{DTW}_r$ aligns each s_i to at least one of the values within $[l_i, u_i]$, i. e.,

$$\mathtt{LB}_{\mathtt{Keogh}}(S,T) \coloneqq \sum_{i=1}^{m} \mathtt{d}\left(s_i, [l_i, u_i]\right)^2 \leqslant \mathtt{DTW}_r(S,T)$$

where

$$\mathtt{d}\left(s_i, [l_i, u_i]\right) \coloneqq \min_{x \in [l_i, u_i]} \mathtt{d}(s_i, x) = \begin{cases} l_i - s_i & \text{iff } s_i < l_i \\ s_i - u_i & \text{iff } u_i < s_i \\ 0 & \text{else} \end{cases} \tag{3.2}$$

The size of the intervals $[l_i, u_i]$ is monotonically decreasing with decreasing Sakoe-Chiba bandwidth r, which will in turn increase $\mathtt{LB}_{\mathtt{Keogh}}$'s tightness to $\mathtt{DTW}_r$. On the other hand, considering a bandwidth of $r = 100\%$, $\mathtt{LB}_{\mathtt{Keogh}}$ is also a lower bound to DTW but its effictiveness degrades to a function comparing the minimum and maximum values of two time series.

The computation of the intervals $[l_i, u_i]$ takes linear time when using the algorithm of Daniel Lemire [64]. Hence, the computation time of $\mathtt{LB}_{\mathtt{Keogh}}$ is linear in the length of the time series as well.

3.3 $\mathtt{LB}_{\mathtt{Keogh}}$ on Multi-dimensional Time Series

Consider two time series $S = (s_1, \cdots, s_m)$ and $T = (t_1, \cdots, t_n)$ with $s_i, t_j \in \mathbb{R}^k$, i. e., S and T are multi-dimensional time series, and let

$$\mathtt{d}(s_i, t_j) \coloneqq \|s_i - t_j\|_2$$

be the Euclidean distance of the two vectors s_i and t_j. In this section, the lower bound $\mathtt{LB}_{\mathtt{Keogh}}$ of $\mathtt{DTW}_r$ is canonically extended using the interpretation presented in Section 3.2.2: Each s_i is assigned an axis aligned minimal bounding box for all vectors $\{t_{i-r}, \cdots, t_{i+r}\}$. This section proves that summing up the shortest square distances of s_i to their bounding boxes is a lower bound for $\mathtt{DTW}_r$ on multi-dimensional time series again.

[2] Note, that Keogh applies the square norm on the warping path [58]. Therefore, $\mathtt{DTW}_r$ with squared distances of the elements is considered here. For readability, the square root in the end of the computation is omitted.

Similar to the lower and upper bounds in Equation (3.1), the following vectors l_i and u_i define a minimal axis aligned bounding box B_i of the values $t_{i-r}, \cdots, t_{i+r}$:

$$B_i \coloneqq \bigotimes_{1 \leqslant j \leqslant k} [l_{i,j}, u_{i,j}] \coloneqq \bigotimes_{1 \leqslant j \leqslant k} \left[\min_{-r \leqslant \theta \leqslant r} t_{i+\theta,j}, \max_{-r \leqslant \theta \leqslant r} t_{i+\theta,j} \right].$$

The distance of a vector to the bounding box is the distance to the nearest element within the bounding box:

$$\begin{aligned} \mathtt{d}\left(s_i, B_i\right)^2 &\coloneqq \min_{t \in B_i} \mathtt{d}\left(s_i, t\right)^2 \\ &= \min_{t \in B_i} \|s_i - t\|_2^2 \\ &= \min_{t \in B_i} \sum_{1 \leqslant j \leqslant k} |s_{i,j} - t_j|^2 \\ &= \sum_{1 \leqslant j \leqslant k} \min_{t_j \in B_{i,j}} |s_{i,j} - t_j|^2 \\ &= \sum_{1 \leqslant j \leqslant k} \mathtt{d}\left(s_i, [l_{i,j}, u_{i,j}]\right)^2 \end{aligned}$$

Considering the definition of $\mathtt{DTW}_r$, the following function is a lower bound for $\mathtt{DTW}_r$:

$$\mathtt{LB}_{\Sigma\mathtt{min}}(S, T) \coloneqq \sum_{i=1}^{m} \min_{-r \leqslant \theta \leqslant r} \mathtt{d}\left(s_i, t_{i+\theta}\right)^2 \leqslant \mathtt{DTW}_r(S, T)$$

The runtime of the distance function $\mathtt{LB}_{\Sigma\mathtt{min}}$ scales with the length of the time series and the width of the Sakoe-Chiba band. To improve the runtime, the following lower bound is used:

$$\begin{aligned} \mathtt{LB}_{\Sigma\mathtt{min}}(S, T) \geqslant \mathtt{LB}_{\mathtt{Box}}(S, T) &\coloneqq \sum_{i=1}^{m} \mathtt{d}\left(s_i, B_i\right)^2 \\ &= \sum_{i=1}^{m} \sum_{j=1}^{k} \mathtt{d}\left(s_i, [l_{i,j}, u_{i,j}]\right)^2 \\ &= \sum_{j=1}^{k} \mathtt{LB}_{\mathtt{Keogh}}\left(S^j, T^j\right) \end{aligned} \tag{3.3}$$

As it turns out, the estimation is not only a canonical extension of the lower bound proposed by Keogh but it can also be computed by using his proposed algorithm on the different dimensions of the time series. In particular, $\mathtt{LB}_{\mathtt{Box}} \equiv \mathtt{LB}_{\mathtt{Keogh}}$ holds for one-dimensional time series.

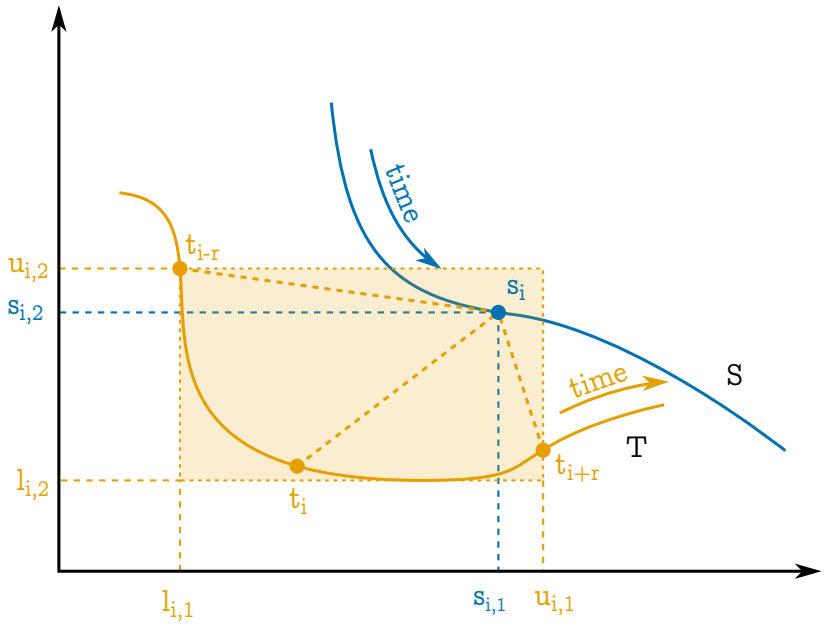


Figure 3.4: Two two-dimensional time series. $\texttt{DTW}_r$ with Sakoe-Chiba Band associates q_i with at least one of $t_{i-r}, \cdots, t_{i+r}$. $\texttt{LB}_{\texttt{Box}}$ compares q_i with the corresponding bounding box $[l_{i,1}, u_{i,1}] \otimes [l_{i,2}, u_{i,2}]$.

Curse of dimensionality: Consider the bounding box of a subsequence $(t_{i-r}, \cdots, t_{i+r})$ as illustrated in Figure 3.4. For two-dimensional time series, the following situation might happen: The time series T moves along the left and then along the bottom edge of the bounding box. The query point s_i is at the top-right vertex of the bounding box. Hence, a perfect alignment of $\texttt{DTW}_r$ (or even DTW) would still result in

$$\texttt{LB}_{\Sigma\texttt{min}}(S,T) = \cdots + \min_{-r \leqslant \theta \leqslant r} \texttt{d}\left(s_i, t_{i+\theta}\right)^2 + \cdots > 0$$

at the alignment of s_i with $t_{i-r}, \cdots, t_{i+r}$. Simplifying the distance function to the bounding box results in

$$\begin{aligned}\texttt{LB}_{\texttt{Box}}(S,T) &= \cdots + \sum_{j=1}^{k} \texttt{d}\left(s_i, [l_{i,j}, u_{i,j}]\right)^2 + \cdots \\ &= \cdots + 0 + \cdots\end{aligned}$$

Thus, there is a clear divergence of $\texttt{DTW}_r$ and $\texttt{LB}_{\texttt{Box}}$. With increasing dimensionality, there is more space for the time series to sneak past the query point, i. e., the probability of this situation increases. For this reason, the *tightness* (i. e., the ratio $\frac{\texttt{LB}_{\texttt{Box}}}{\texttt{DTW}_r}$) of the lower bound gets worse with increasing dimensionality, and therefore fewer computations of the expensive $\texttt{DTW}_r$

function can be pruned in nearest neighbor queries. This effect is similar to that of the *Curse of Dimensionality* for metric index structures [31].

The goal of this section is the proof of Theorem 2 that claims the existence of the Curse of Dimensionality on the lower bound $\texttt{LB}_{\texttt{Box}}$ to $\texttt{DTW}_r$.

Theorem 2. Let S and T be two time series in $\mathbb{R}^k$ with length n and independent but identically distributed elements. Then

$$\frac{\mathbb{E}\left[\texttt{LB}_{\texttt{Box}}(S,T)\right]}{\mathbb{E}\left[\texttt{DTW}_r(S,T)\right]} \longrightarrow 0$$

for $k, n \to \infty$.

The Curse of Dimensionality does not affect trivial datasets, e. g., when the time series only move within one dimension, then the dataset effectively remains one-dimensional, although the representational dimensionality may increase arbitrarily. Hence, proof for the existence of the Curse of Dimensionality requires non-trivial datasets. Therefore, Theorem 2 studies the Curse of Dimensionality on finite datasets of time series with independent but identically distributed elements. This assumption makes the proof feasible while presuming as little knowledge about the dataset as possible. Although this is not a realistic assumption (e. g., because succeeding elements of a time series are usually close by each other), Section 3.5 confirms the theoretical results experimentally on real-world datasets.

The following lemmas are used in the proof of Theorem 2.

Lemma 5. Let $D_{\theta,j}$ be independent but identically distributed random variables for $1 \leqslant \theta \leqslant r$ and $1 \leqslant j \leqslant k$.[3] Furthermore, let $\mu_r \coloneqq \mathbb{E}\left[\min_{1\leqslant\theta\leqslant r} D_{\theta,j}\right]$ and $\mu \coloneqq \mathbb{E}\left[D_{\theta,j}\right]$. Then,

$$\frac{\mathbb{E}\left[\sum_{j=1}^{k} \min_{1\leqslant\theta\leqslant r} D_{\theta,j}\right]}{\mathbb{E}\left[\min_{1\leqslant\theta\leqslant r} \sum_{j=1}^{k} D_{\theta,j}\right]} \longrightarrow \frac{\mu_r}{\mu} \quad \text{for } k \to \infty. \tag{3.4}$$

Proof. Let $\mu \coloneqq \mathbb{E}\left[D_{\theta,j}\right]$ be the mean and $\sigma^2 \coloneqq \mathbb{V}\left[D_{\theta,j}\right]$ be the variance of the identically distributed variables $D_{\theta,j}$. The following inequation holds using calculation rules for expected values:

$$\mathbb{E}\left[\sum_{j=1}^{k} \min_{1\leqslant\theta\leqslant r} D_{\theta,j}\right] = k \cdot \mathbb{E}\left[\min_{1\leqslant\theta\leqslant r} D_{\theta,1}\right] = k \cdot \mu_r \tag{3.5}$$

[3] Remember, that this section considers time series in $\mathbb{R}^k$.

To estimate the denominator, first the Markov inequality as well as some basic transformations are applied:

$$\begin{aligned}
\mathbb{E}\left[\min_{1\leqslant\theta\leqslant r}\sum_{j=1}^{k} D_{\theta,j}\right] &\geqslant a\cdot\mathbb{P}\left[\min_{1\leqslant\theta\leqslant r}\sum_{j=1}^{k} D_{\theta,j}\geqslant a\right]\\
&\geqslant a\cdot\mathbb{P}\left[\bigwedge_{1\leqslant\theta\leqslant r}\sum_{j=1}^{k} D_{\theta,j} > a\right]\\
&= a\cdot\mathbb{P}\left[\sum_{j=1}^{k} D_{1,j} > a\right]^r\\
&= a\cdot\left(1-\mathbb{P}\left[\sum_{j=1}^{k} D_{\theta,j}\leqslant a\right]\right)^r\\
&= a\cdot\left(1-\mathbb{P}\left[\frac{\sum_{j=1}^{k} D_{1,j}-k\cdot\mu}{\sigma\sqrt{k}}\leqslant\frac{a-k\cdot\mu}{\sigma\sqrt{k}}\right]\right)^r \qquad (3.6)
\end{aligned}$$

Now, let $\rho \coloneqq \mathbb{E}\left[\left|D_{\theta,j}^3\right|\right]$. Since only finite datasets are considered for the nearest neighbor queries, $\rho < \infty$ can safely be assumed. For a theoretical analysis on datasets with an infinite number of elements, this property is a necessary preliminary for the Berry-Esseen Theorem[4], which is used for the further estimation of the denominator from Equation (3.6):

$$\begin{aligned}
\mathbb{E}\left[\min_{1\leqslant\theta\leqslant r}\sum_{j=1}^{k} D_{\theta,j}\right] &\geqslant a\cdot\left(1-\mathbb{P}\left[\frac{\sum_{j=1}^{k} D_{1,j}-k\cdot\mu}{\sigma\sqrt{k}}\leqslant\frac{a-k\cdot\mu}{\sigma\sqrt{k}}\right]\right)^r\\
&\geqslant a\cdot\left(1-\left(\Phi_{0,1}\left(\frac{a-k\cdot\mu}{k\sqrt{k}}\right)+\frac{C\cdot\rho}{\sigma^3\cdot\sqrt{k}}\right)\right)^r\\
&= k\cdot\mu\cdot\left(1-\left(\frac{C\cdot\rho}{\sigma^3\cdot\sqrt{k}}\right)\right)^r \text{ for } a=k\cdot\mu \qquad (3.7)
\end{aligned}$$

The Berry-Esseen Theorem used in Inequation (3.7) also claims the existence of the constant C, which is independent of k.

Using Equation (3.5) and Inequation (3.7) yields the desired convergence:

$$\frac{\mathbb{E}\left[\sum_{j=1}^{k}\min_{1\leqslant\theta\leqslant r} D_{\theta,j}\right]}{\mathbb{E}\left[\min_{1\leqslant\theta\leqslant r}\sum_{j=1}^{k} D_{\theta,j}\right]}\leqslant\frac{k\cdot\mu_r}{k\cdot\mu\cdot\left(1-\left(\frac{C\cdot\rho}{\sigma^3\cdot\sqrt{k}}\right)\right)^r}\longrightarrow\frac{\mu_r}{\mu}\text{ for } k\to\infty$$

□

[4]The Berry-Esseen Theorem claims that the sum of random variables converges to a normal distribution for increasing number of summands [18].

For $r = R \cdot n$ $(0 < R \leqslant 1)$, the following Lemma 6 shows that the value $\frac{\mu_r}{\mu}$ from Lemma 5 converges to 0 for $n \to \infty$.

Lemma 6. Let $X_i \geqslant 0$ be independent but identically distributed random variables for $1 \leqslant i \leqslant n$, such that $\mathbb{P}[X_i \geqslant x] < 1$ for each $x > 0$. Then, $\mathbb{E}[\min_{1 \leqslant i \leqslant n} X_i] \longrightarrow 0$ for $n \to \infty$.

Proof. The expected value can be calculated as follows:

$$\begin{aligned}
\mathbb{E}\left[\min_{1 \leqslant i \leqslant n} X_i\right] &= \int_0^\infty \mathbb{P}\left[\min_{1 \leqslant i \leqslant n} X_i \geqslant x\right] dx \\
&= \int_0^\infty \mathbb{P}\left[X_1 \geqslant x \wedge \cdots \wedge X_n \geqslant x\right] dx \\
&= \int_0^\infty \mathbb{P}\left[X_i \geqslant x\right]^n dx
\end{aligned}$$

Since $\mathbb{P}[X_i \geqslant x]$ is Lebesgue-measurable and $\mathbb{P}[X_i \geqslant x]^n \longrightarrow 0$ for $x > 0$, the last integral converges to zero for $n \to \infty$. □

Lemma 5 and 6 are used to prove that $\mathtt{LB}_{\mathtt{Box}}$ suffers from the curse of dimensionality.

Proof of Theorem 2 on random time series. As mentioned earlier, it is assumed that the time series elements are independent but identically distributed random variables. Therefore, the distances between any two elements

$$D_{i,j} \coloneqq \mathtt{d}(s_i, t_j)^2$$

are independent but identically distributed random variables as well. Let $r = R \cdot n$ ($0 < R \leqslant 1$ constant) be the width of the Sakoe-Chiba band. Then, the claimed convergence follows using the definition of $\mathtt{LB}_{\mathtt{Box}}$ and $\mathtt{LB}_{\Sigma\mathtt{min}}$ as well as Lemma 5 and 6:

$$\begin{aligned}
\frac{\mathbb{E}[\mathtt{LB}_{\mathtt{Box}}(S,T)]}{\mathbb{E}[\mathtt{DTW}_r(S,T)]} &\leqslant \frac{\mathbb{E}\left[\sum_{i=1}^n \sum_{j=1}^k \mathtt{d}(s_i, [l_{i,j}, r_{i,j}])^2\right]}{\mathbb{E}[\mathtt{LB}_{\Sigma\mathtt{min}}(S,T)]} \\
&\leqslant \frac{\mathbb{E}\left[\sum_{i=1}^n \sum_{j=1}^k \min_{-r \leqslant \theta \leqslant r} D_{i+\theta,j}\right]}{\mathbb{E}\left[\sum_{i=1}^n \min_{-r \leqslant \theta \leqslant r} \sum_{j=1}^k D_{i+\theta,j}\right]} \\
&= \frac{n \cdot \mathbb{E}\left[\sum_{j=1}^k \min_{1 \leqslant \theta \leqslant 2r+1} D_{\theta,j}\right]}{n \cdot \mathbb{E}\left[\min_{1 \leqslant \theta \leqslant 2r+1} \sum_{j=1}^k D_{\theta,j}\right]} \\
&\longrightarrow \frac{\mu_r}{\mu} \quad \text{for } k \to \infty \\
&\longrightarrow 0 \quad \text{for } n \to \infty
\end{aligned}$$

□

3.4 Algorithms for the Dog Keeper Distance

Practical implementations of DTW (with and without a Sakoe-Chiba band constraint) speed up the computation process by stopping early when a *promised lower bound* already exceeds a certain threshold. For example, when processing the (constrained) warping matrix while computing DTW, the minimum value of a column or row is a lower bound for the final distance value.

Since DTW sums up values along a warping path through the distance matrix, the values of the warping matrix in the early computation time exceed a certain threshold less probable than the values at a later computation time. This observation does not hold when computing the DK distance [9]. This insight yields the idea that the matrix filled out during computation of the DK distance might be very sparse regarding cells that exceed the threshold.

Therefore, the approach followed in this section improves the computation time of the DK distance by computing the distance matrix using a *sparse matrix* algorithm (cf. Section 3.4.2). A low threshold is necessary to avoid computing most of the cells of the distance matrix. Such a threshold is found using a cheap upper bound for the DK distance. Specifically, Section 3.4.1 proposes a greedy algorithm to the time warping alignment problem.

3.4.1 Greedy Dog Keeper

Consider two arbitrary but fixed time series $S = (s_1, \cdots, s_m)$ and $T = (t_1, \cdots, t_n)$. The greedy Dog Keeper distance (GDK) starts by aligning s_1 and t_1. It then successively steps to one of the next pairs aligning (s_{i+1}, t_j), (s_i, t_{j+1}), or (s_{i+1}, t_{j+1}) with the lowest distance. When it reaches the alignment of s_m to t_n, the maximum distance value along the chosen path yields the final distance value. Algorithm 8 provides the pseudo-code for the algorithm.

Algorithm 8 Greedy Dog Keeper distance

```
1  Algorithm: greedydogkeeper
2  Input: time series S,T of length m,n, resp.; threshold ε
3  Output: upper bound d
4
5  let (i,j) = (1,1)
6  let g = d(s_i, t_j)
7  while i ≠ m and j ≠ n and g ⩽ ε
8     // non defined distances yield ∞
9     (i,j) = arg min {d(s_{i+1}, t_j), d(s_{i+1}, t_{j+1}), d(s_i, t_{j+1})}
10    g = max {g, d(s_i, t_j)}
11 return g
```

Since GDK does not necessarily choose the optimal warping path, it is an upper bound to DK. The application of a Sakoe-Chiba band is possible as well but left out for the sake of readability.

The GDK distance matches whole sequences against each other. In order to support subsequence matching, alter the algorithm as follows: First, find the best match of s_1 to any t_i, i. e., $t_{left} := \arg\min_{1 \leqslant i \leqslant n} \mathtt{d}(s_1, t_i)$. Then, let GDK start its computation at t_{left} and stop the computation as soon as s_m is aligned to one of the t_i. For details, call for Algorithm 9.

Algorithm 9 Greedy Dog Keeper for subsequence search

```
1  Algorithm: greedydogkeeper
2  Input: time series S and T of length m and n, resp.;
       threshold ε
3  Output: upper bound d
4
5  let (i, j) = (1, arg min_{1⩽j⩽n} {d(s_1, t_j)})
6  let g = d(s_i, t_j)
7  while i ≠ m
8     // non defined distances yield ∞
9     (i, j) = arg min {d(s_{i+1}, t_j), d(s_{i+1}, t_{j+1}), d(s_i, t_{j+1})}
10    g = max {g, d(i, j)}
11    if g > ε then return g
12 return g
```

Both algorithms have linear complexity since the while loop runs at maximum $m + n$ times.

3.4.2 Sparse Dog Keeper Distance

The sparse Dog Keeper algorithm essentially works the same as the original algorithm, except that it only visits those neighbor cells of the distance matrix with a value not larger than a given threshold. Algorithm 10 provides the pseudo-code for the algorithm: Similar to the original algorithm, it computes the (sparse) warping matrix column by column (cf. Line 10). The variables I and J store the cells' indices to visit in the current and successor columns. If the value at the current cell of the matrix is not larger than the threshold (cf. Line 13), the right, upper right, and upper cells also need to be visited (cf. Lines 15 and 16). The columns' actual values are stored in D (current column) and E (previous column). After finishing the column's computation, the variables to enter the next column are prepared (cf. Lines 19 to 26).

The algorithm performs subsequence search iff passing `true` for the parameter SUB. It differs from the whole matching version by considering each

Algorithm 10 Sparse Dog Keeper distance

```
1  Algorithm: sparsedogkeeper
2  Input: time series S and T of length m and n, resp.;
       threshold ε; boolean SUB
3  Output: distance d
4
5  I = {1}
6  J = ∅
7  D = (∞, ···)
8  E = (∞, ···)
9
10 for k = 1 to n
11     if SUB then d = ∞
12     for i ∈ I
13         if d(S_i, T_k) ⩽ ε
14             D_i = max{d(S_i, T_k), min{D_{i−1}, E_i, E_{i−1}}}
15             I = I ∪ {i + 1}
16             J = J ∪ {i, i + 1}
17         else
18             I = I \ {i}
19     if SUB then d = min{d, D_m}
20     E = D
21     // reset D to (∞, ···)
22     for i ∈ I
23         D_i = ∞
24     if SUB then I = J ∪ {1}
25     else I = J
26     J = ∅
27     if not SUB and I is empty
28         return ε
29 if SUB then return d
30 else return D_m
```

position of the super sequence as the possible start of a match (cf. Line 24) and considering each position as the possible end of a match (cf. Line 19).

3.4.3 Nearest Neighbor Search

The nearest neighbor search algorithm that uses the sparse DK distance requires a good upper bound as a threshold. Hence, the first step is to loop over all time series and find the lowest upper bound using the GDK distance. A standard nearest neighbor search is then performed by scanning the time series using the threshold of the GDK distance.

3.5 Evaluation

Section 3.5.1, experimentally evaluates Theorem 2 on both, synthetic datasets (cf. Section 2.5) and real world datasets [67]. To that, $\texttt{DTW}_r$ (with $r = 10\%$ and $\texttt{LB}_{\texttt{Box}}$ as lower bound) is compared against the proposed sparse DK distance in terms of runtime in Section 3.5.2. It is shown that DK outperforms DTW on high dimensional time series. Only subsequence matching algorithms are considered as this is the more challenging task and yields results for whole matching algorithms. Since $\texttt{DTW}_r$ and DK slightly differ in their semantics, both distance functions are compared on classification tasks using one-nearest neighbor classifiers in Section 3.5.3. They are shown to yield similar accuracy.

Synthetic data sets: Results are not presented for all parameters of the dataset generators, but the following set of parameters yield the most interesting results: If not mentioned otherwise, the time series are generated with distortion 25, radius 50, 50 distinct classes, and 2 representatives per class using the RAM generator (i. e., 100 time series per dataset) and 27 distinct classes with 3 representatives per class using the CBF generator (i. e., 81 time series per data set).

Implementation: The implementation of $\texttt{LB}_{\texttt{Box}}$ is based on $\texttt{LB}_{\texttt{Keogh}}$ from the UCR Suite [78]. The time series' normalization is skipped on some datasets to improve the runtime and accuracy of $\texttt{LB}_{\texttt{Box}}$. While adapting the UCR Suite to work on multi-dimensional time series, the runtime has been verified to remain stable for one-dimensional time series; thus, no result for runtime comparison is implementation-dependent. All experiments were executed on one core of an Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00GHz with 24GB Memory.

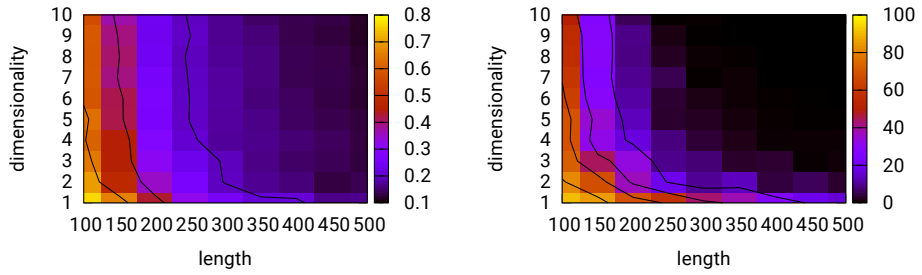


Figure 3.5: Heat maps presenting the average tightness of $\texttt{LB}_{\texttt{Box}}$ (left) and the pruning power of $\texttt{LB}_{\texttt{Box}}$ in percent (right) on the `RAM` dataset.

Experiments: For each dataset, a one-nearest neighbor query has been performed as a linear scan with both `DK` and $\texttt{DTW}_r$. In the latter case, the runtime of the queries improved with the lower bound $\texttt{LB}_{\texttt{Keogh}}$ [78]. The runtime and the classification scores of each query have been measured and recorded. Also, the pruning power for the nearest neighbor queries with $\texttt{DTW}_r$ and $\texttt{LB}_{\texttt{Box}}$ has been recorded, i. e., it has been counted how many time series could be pruned with the result of the lower bound, already. The real-world datasets consist of a training and a test data set. A one-nearest neighbor query has been performed for each test dataset element against the complete training dataset. On the synthetic datasets, leave-one-out cross-validation has been performed, i. e., a one-nearest neighbor query was executed for each element of a dataset against the rest of the dataset, respectively.

3.5.1 Curse of dimensionality:

Theorem 2 claims that the tightness of the lower bound function $\texttt{LB}_{\texttt{Box}}$ to $\texttt{DTW}_r$ gets worse with increasing dimensionality and length of the time series. This theorem is evaluated experimentally on datasets generated by the two synthesizers `CBF` and `RAM` (cf. Section 2.5).

Tightness: Figure 3.5 demonstrates that the tightness of $\texttt{LB}_{\texttt{Box}}$ to $\texttt{DTW}_r$ is decreasing down to zero for increasing length, which confirms Theorem 2. The theorem also claims that the tightness drops to a constant value for increasing dimensionality but constant length. With Figure 3.5 this claim is confirmed for the `RAM` datasets. It is shown even more that the tightness converges (drops down) already for moderate dimensionality (e. g., three-dimensional time series). The same results can be observed on the `CBF` datasets (cf. Figure 3.6) although the dimensionality has a larger impact on the tightness while the length has a smaller impact.

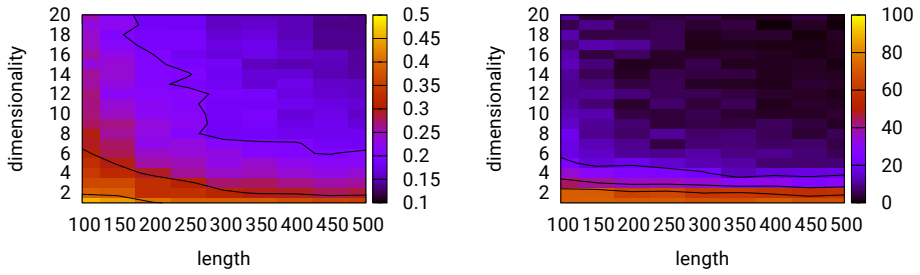


Figure 3.6: Heat maps presenting the average tightness of $\texttt{LB}_{\texttt{Box}}$ (left) and the pruning power of $\texttt{LB}_{\texttt{Box}}$ in percent (right) on the `CBF` dataset.

Pruning power: The right hand side heat maps of Figures 3.5 and 3.6 show the effects of the dimensionality and length of the time series on the pruning power, respectively. They confirm the correlation between the dropping of the tightness of $\texttt{LB}_{\texttt{Box}}$ and the pruning power's dropping in nearest neighbor queries.

As can be seen, the dimensionality has a larger impact on the pruning power than it has on the tightness of the time series. The reason is the curse of dimensionality on retrieval tasks [31], which claims that the variance of distance values between random elements of a dataset decreases with increasing dimensionality. In this case, even if the tightness of a lower bound such as $\texttt{LB}_{\texttt{Box}}$ equals on two distinct datasets, pruning is still less probable in the higher dimensional dataset. Figure 3.7 illustrates an example for this effect: Here, q is the query, y is the nearest neighbor found so far during a search, and x is the next candidate element of the data set. Assume that $l(q,x) = \alpha \cdot d(q,x)$ for $0 \leqslant \alpha \leqslant 1$ holds in both, the low-dimensional and the high-dimensional dataset, i. e., the tightness of the lower bound function equals in both datasets. In the lower-dimensional dataset, the lower bound $l(q,x)$ to the candidate is larger than the best distance $d(q,y)$ found so far, and thus the next element x can be pruned from the search. In the higher-dimensional dataset, the variance of distances is smaller, and thus the best distance $d(q,y)$ found so far is probably larger than the lower bound $l(q,x)$ to the candidate, in which case x needs to be examined closer by computing the expensive distance $d(q,x)$. Hence, since $d(q,x)$ and $d(q,y)$ converge on datasets with increasing dimensionality (this is the curse of dimensionality), $l(q,x) < d(q,y)$ is more probable with higher dimensionality, and thus the pruning power decreases. Note that this insight holds for any lower bound driven approaches.

Both Figures 3.5 and 3.6 show that $\texttt{LB}_{\texttt{Box}}$ loses its advantage already for moderate dimensionality.

For the evaluation of the runtime of the sparse `DK` algorithm, the runtime of $\texttt{DTW}_r$ with $\texttt{LB}_{\texttt{Box}}$ is considered as the baseline. Figures 3.8 and 3.9 show the

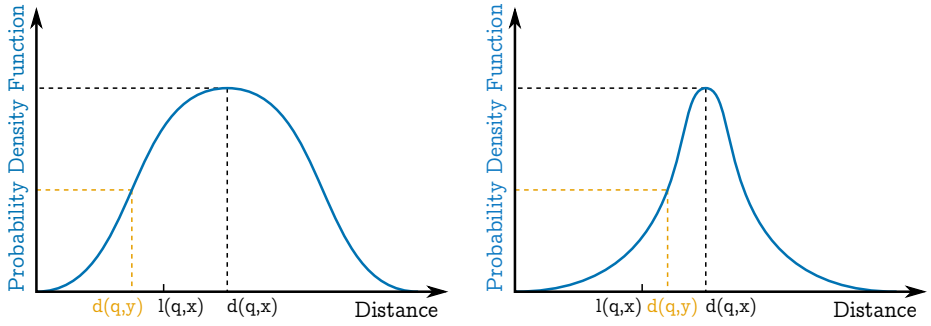


Figure 3.7: Sketch of the distance distribution of two datasets (left: low-dimensional; right: high-dimensional), a query q, a nearest neighbor y, and a candidate x.

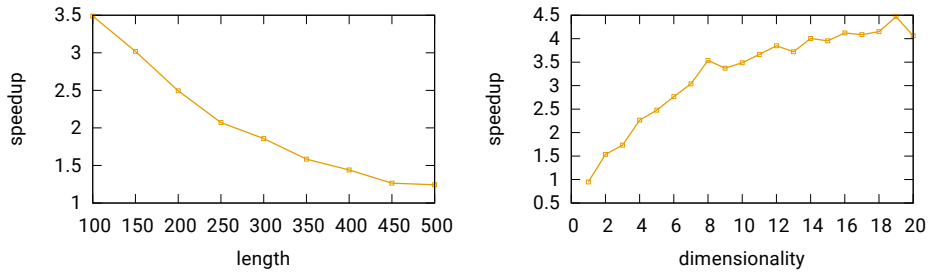


Figure 3.8: Speedup of DK to $\texttt{DTW}_r$ with $\texttt{LB}_{\texttt{Box}}$ on the CBF data set. Dimensionality (left): 10; Length (right): 100.

speedup of the sparse DK implementation against $\texttt{DTW}_r$ with $\texttt{LB}_{\texttt{Box}}$ on the RAM and CBF datasets, respectively. The speedup decreases for longer time series while increasing with growing dimensionality. Figure 3.10 confirms this observation on an even larger parameter set. Thus, the sparse DK implementation outperforms $\texttt{DTW}_r$ with $\texttt{LB}_{\texttt{Box}}$ on rather short and multi-dimensional time series.

3.5.2 Computation Time

The same experiments have been repeated on the following real-world data sets from the UCI Machine Learning Repository [67] to ensure that the results do not depend on the generated synthetic datasets: Character Trajectories (CT), Activity Recognition system based on Multi-sensor data fusion (AReM) [77], EMG Physical Action (EMGPA), Australian Sign Language 2 (ASL) [56], Arabic Spoken Digits (ASD), and Vicon Physical Action (VICON). The ECG data proposed by Keogh [78] for querying in one very long time series has been examined. Columns 1 and 2 of Table 3.1 show that the speedup increases with growing dimensionality. Again, this demonstrates that the sparse DK distance outperforms $\texttt{DTW}_r$ with $\texttt{LB}_{\texttt{Box}}$ in terms of

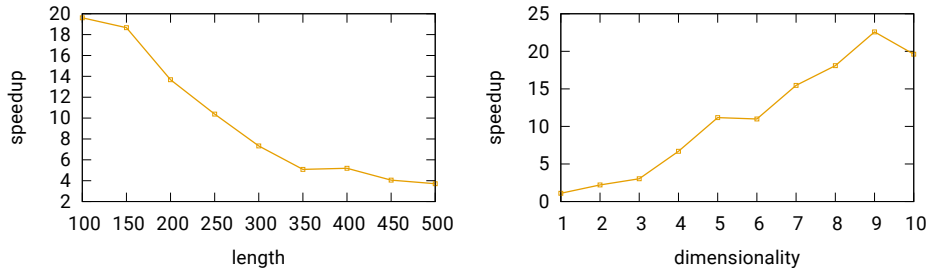


Figure 3.9: Speedup of DK to $\texttt{DTW}_r$ with $\texttt{LB}_{\texttt{Box}}$ on the RAM data set. Dimensionality (left): 10; Length (right): 100.

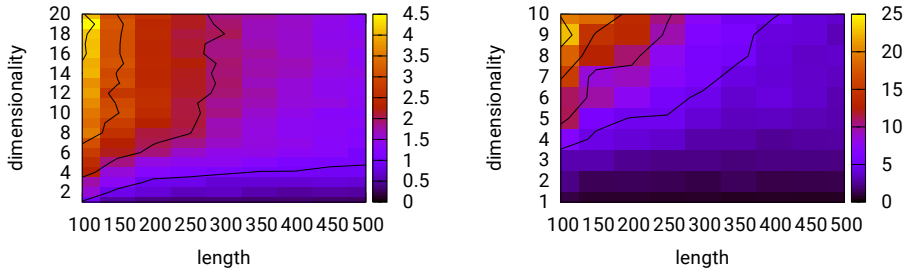


Figure 3.10: Speedup of DK to $\texttt{DTW}_r$ with $\texttt{LB}_{\texttt{Box}}$ on the CBF data set (left) and the RAM data set (right). Dimensionality (left): 10; Length (right): 100.

Table 3.1: Speedup and Accuracy of $\texttt{DTW}_r$ with $\texttt{LB}_{\texttt{Box}}$ and DK on real world data sets.

data set	dim.	speedup	acc. $\texttt{DTW}_r$	acc. DK
ECG	1	0.05	-	-
CT	2	4.5	0.94	0.93
AReM (without Z-Normalization)	6	1.2	0.81	0.75
EMGPA	8	31	0.21	0.26
ASD	13	45	0.98	0.96
ASL	22	33	0.85	0.88
VICON	27	62	0.12	0.09

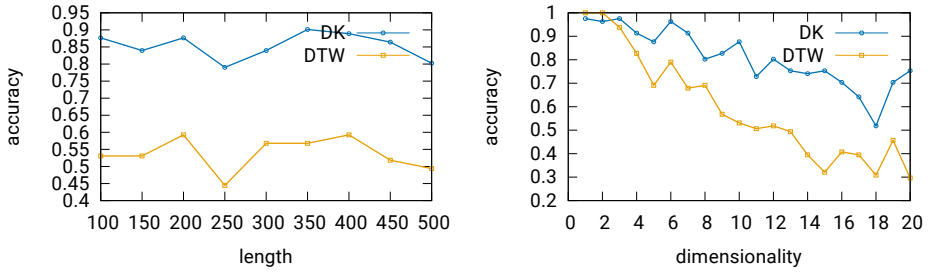


Figure 3.11: Accuracy of DK and $\texttt{DTW}_r$ with $\texttt{LB}_{\texttt{Box}}$ on the CBF data set. Dimensionality: 10 (left); Length: 100 (right).

computation time especially on multi-dimensional time series.

3.5.3 Retrieval Quality

Figures 3.11 and 3.12 reveal that the accuracy of both distance functions, DK and $\texttt{DTW}_r$, decreases on CBF while increasing on RAM with growing dimensionality. While DK outperforms $\texttt{DTW}_r$ on the CBF data set it looses on the RAM data set.

Columns 1, 3, and 4 of Table 3.1 also show that there is no clear winner regarding accuracy. Hence, $\texttt{DTW}_r$ and DK seem to be exchangeable when it comes to the semantics of comparisons.

3.6 Conclusion

This chapter introduced the lower bound $\texttt{LB}_{\texttt{Box}}$ as a canonical extension of Keogh's lower bound $\texttt{LB}_{\texttt{Keogh}}$ to DTW with Sakoe-Chiba band ($\texttt{DTW}_r$) on multi-dimensional time series, including a proof for its correctness. Not only do lower bounds suffer from the curse of dimensionality even if their tightness remains constant, but it was also proven that the tightness of $\texttt{LB}_{\texttt{Box}}$ decreases

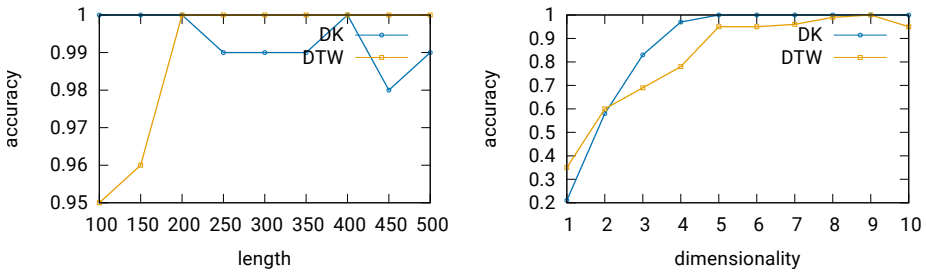


Figure 3.12: Accuracy of DK and $\texttt{DTW}_r$ with $\texttt{LB}_{\texttt{Box}}$ on the RAM data set. Dimensionality: 10 (left); Length: 100 (right).

with increasing length and dimensionality of the time series. On the other hand, this chapter presented and proposed an alternative algorithm for the computation of the long-known DK distance, which is similar to $\texttt{DTW}_r$ in various aspects (e. g., its semantic and the formal definition). Note that the DK distance satisfies the triangle inequality and thus is applicable in metric indexes (cf. Section 2.4).

The evaluation in this chapter confirmed the theory that $\texttt{LB}_{\texttt{Box}}$, an extension of $\texttt{LB}_{\texttt{Keogh}}$, suffers from the curse of dimensionality. The proposed sparse DK implementation was shown to outperform $\texttt{DTW}_r$ with $\texttt{LB}_{\texttt{Box}}$ on multi-dimensional synthesized datasets and real-world datasets in computation time by more than one order of magnitude. On the other hand, $\texttt{DTW}_r$ and DK seem to be exchangeable since there is no clear winner regarding retrieval tasks' accuracy. Hence, regarding time-warping distance functions, this thesis concludes with the proposition to stay with $\texttt{LB}_{\texttt{Keogh}}$ on one-dimensional time series while choosing the DK distance on multi-dimensional time series.

Chapter 4

Congruence Distance

Quite some research is devoted to comparing and indexing time series. Chapter 3 considers the comparison of functions that are not affected by time-warping. However, for many application scenarios (e. g., motion gesture recognition in virtual reality), the invariance under isometric spatial transformations (i. e., rotation, translation, and reflection) is as crucial as the invariance under time-warping. Distance functions on time series that are invariant under isometric transformations can be seen as a measurement for the congruence of two time series. The congruence distance proposed in this chapter is the canonical example of such a distance function.

Section 4.2 proves that there is no fast algorithm computing the congruence distance. Several approximations for the congruence distance are developed in Section 4.3 to overcome this issue. Two of these approximations even satisfy the triangle inequality and thus can be used with metric index structures. Section 4.3 also shows that the presented approximations serve as a lower bound for the congruence distance. The evaluation in Section 4.4 shows that they achieve remarkable tightness while providing a speedup of more than two orders of magnitude to the congruence distance.

4.1 Introduction

The distance functions defined and analyzed in this chapter measure the (approximate) congruence of two time series. Thereby, the distance between two time series S and T shall be 0 iff S can be transformed into T by rotation, translation, and reflection; in this case, S and T are said to be *congruent*. A value greater than 0 shall correlate to the amount of transformation necessary to turn the time series into congruent ones.

The classical Congruence problem determines whether two point sets $A, B \subseteq \mathbb{R}^k$ are congruent considering isometric transformations (i. e., rotation, translation, and reflection) [5, 45]. For 2- and 3-dimensional spaces, there are results providing algorithms with runtime $\mathcal{O}(n \cdot \log n)$ when n is

the size of the sets; for larger dimensionalities, an algorithm with runtime $\mathcal{O}(n^{k-2} \cdot \log n)$ exists [5]. For various reasons (e. g., bounded precision of floating-point numbers or physical measurement errors), the *approximated* Congruence problem is of much more interest in practical applications. Different variations of the approximated Congruence problem have been studied (e. g., regarding the types of transformations that are used; whether the assignment of points from A to B is known; regarding the applied metric) [4, 5, 45, 50].

The Congruence problem is related to this work since the problem is concerned with the existence of isometric functions that map point sets to other point sets. The main difference is that this thesis considers ordered lists of points (i. e., time series) rather than sets. As shown in this chapter, solving the approximated Congruence problem is NP-hard regarding the length and dimensionality (cf. Section 4.2).

This chapter contributes by analyzing the congruence distance in Section 4.2 and evaluating it with an implementation based on a nonlinear optimizer in Section 4.4. Several approximations to the congruence distance are proposed in Section 4.3. Some of them are (pseudo) metric distance functions; some have quasi-linear runtime regarding the time series' length. All of the proposed approximations have linear runtime regarding the dimensionality of the time series' elements. The approximations are evaluated experimentally in Section 4.4.

Since computational geometry inspired this topic, this chapter only considers time series in Euclidean vector spaces. Still, the results might be transferred to and examined in other metric spaces (e. g., graphs with the shortest path distance).

4.1.1 Basic Notation

This chapter introduces additional notations. The set of all powers of two is denoted by $2^{\mathbb{N}} \coloneqq \{1, 2, 4, 8, \cdots\}$. The identity matrix is denoted by I and the transposed of a matrix M by M^T. For a matrix M in $\mathbb{R}^k$, the matrix holding the absoloute values is denoted by $|M| \coloneqq (|M_{i,j}|)_{0 \leqslant i,j < k}$. In this chapter, indices of the sequences' elements start with 0 for convenience reasons, e. g., $S = (s_0, \cdots, s_{n-1})$ is a sequence of length n. The abbreviation $\delta_p \coloneqq \|\mathtt{d}_2\|_p$ is used, i. e., $\delta_p(S, T) = \|(\mathtt{d}_2(s_i, t_i))_{0 \leqslant i < n}\|_p$ for two time series $S = (s_0, \cdots, s_{n-1})$ and $T = (t_0, \cdots, t_{n-1})$.

4.1.2 Congruence Distance

This chapter considers the metric space $\mathbb{M} \coloneqq \mathbb{R}^k$ and $\mathcal{T} \coloneqq \mathbb{M}^n$ for arbitrary but fixed $k, n \in \mathbb{N}$. The following definition formally declares the exact congruence of two time series (cf. Section 2.4.2).

Definition 5 (Congruence of time series)**.** Consider the Euclidean metric space $(\mathbb{R}^k, \mathtt{d}_2)$. Two time series S and T of the same length are called *congruent*, which $S \cong_C T$ denotes, if they can be transformed into each other by rotation, translation, and reflection, i. e.,

$$S \cong_C T \iff \exists M \in \mathcal{MO}(k)\ \exists v \in \mathbb{M} :\ T = M \cdot S + v.$$

It is easy to see that for each $n \in \mathbb{N}$, the congruence relation $\cong_C$ is an *equivalence relation* on the set of all time series over $\mathbb{R}^k$ of length n.

This thesis aims at a distance measure that regards two time series S and T as close to each other if they are congruent to a certain degree, particularly that meets the following *congruence requirement*.

Definition 6 (Congruence Requirement)**.** Let $\mathcal{T} = \mathbb{M}^n$. A function $\delta : \mathcal{T} \times \mathcal{T} \longrightarrow \mathbb{R}_{\geqslant 0}$ meets the *congruence requirement* iff for all time series $S, T \in \mathcal{T}$ the equivalence

$$\delta(S, T) = 0 \iff S \cong_C T.$$

holds.

The following example highlights some intuition for the congruence distance function that is provided in Definition 7.

Example 5. Consider the time series

$$S \coloneqq \left(\begin{pmatrix}-4\\0\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}\right) \qquad \text{and} \qquad T \coloneqq \left(\begin{pmatrix}0\\3\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}\right).$$

Obviously, $\delta_1(S, T) = 5$. Rotating T counterclockwise by 90 degrees, i. e., computing $M \cdot T$ for the matrix

$$M \coloneqq \begin{pmatrix}0 & -1\\1 & 0\end{pmatrix}$$

yields

$$M \cdot T = \left(\begin{pmatrix}-3\\0\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}\right)$$

and $\delta_1(S, M \cdot T) = 1 + \sqrt{2} < 5$.

Thus without rotation, a vector of length 5 needs to be added to the first state of T to transform T into S. However, after rotating T by 90 degrees counterclockwise, only a vector of length 1 and a vector of length $\sqrt{2}$ needs to be added to the first and third state of $M \cdot T$, respectively, to obtain the time series S.

Adding vectors to the time series' individual states can be interpreted as investing energy to make both time series have the same structure, i.e., being congruent. Hence, the congruence distance defined below can be interpreted as a measure for the minimum amount of energy necessary to make both time series congruent.

Definition 7 (Congruence Distance)**.** Let $p \in \mathbb{R}_{\geqslant 1}$. The *congruence distance* $\delta_p^C(S,T)$ of two time series $S, T \in \mathcal{T}$ is

$$\delta_p^C(S,T) \coloneqq \min_{M \in \mathcal{MO}(k),\, v \in \mathbb{R}^k} \delta_p\left(S,\, M \cdot T + v\right).$$

Similarities to time-warping distance functions: The distance function DTW is (nearly) invariant under time-warping while comparing two time series S and T (cf. Chapter 3). In detail, considering $\sigma(S)$ and $\tau(T)$ as time-warping by duplicating elements (e. g., $\sigma(S) = (s_0, s_1, s_1, s_2, s_3, s_3, \cdots)$) yields that DTW minimizes the L1 distance ¸over all time warps:

$$\texttt{DTW}(S,T) \coloneqq \min_{\sigma,\tau} \|\mathtt{d}\left(\sigma(S), \tau(T)\right)\|_1 \quad \text{with } \#\sigma(S) = \#\tau(T).$$

Analogously, this statement holds for the DK distance.

On the other hand, the congruence distance is invariant under all isometric transformations. The difference to DTW is that it minimizes the L1 distance by multiplying an orthogonal matrix and adding a vector:

$$\delta_1^C(S,T) \coloneqq \min_{M,v} \|\mathtt{d}\left(S, M \cdot T + v\right)\|_1$$

Congruence distance as an optimization problem: Considering fixed time series S and T, the computation of $\delta_1^C(S,T)$ is an optimization problem

$$\delta_1^C(S,T) \coloneqq \min_{M,v}\ f(M,v)$$

where $f \coloneqq \sum_{i=0}^{n-1} \mathtt{d}_2\left(s_i, M \cdot t_i + v\right)$ corresponds to the objective function. For time series in $\mathbb{R}^k$, the orthogonality of M yields a set of k^2 equality-based constraints:

$$\langle m_i, m_j \rangle = \begin{cases} 1 & \text{iff } i = j \\ 0 & \text{else} \end{cases}$$

4.2 Time Series Congruence

Section 4.2.1 studies properties of the congruence distance: The results are that the congruence distance is well defined, invariant under isometric transformations, and a pseudo metric distance function. In Section 4.2.2, the congruence distance is shown to be hard to compute.

4.2.1 Properties of the Congruence Distance

Although $\mathcal{MO}(k)$ and $\mathbb{R}^k$ are infinite sets, the following lemma shows that the "min" used in Definition 7 of the congruence distance δ_p^C does exist, i. e., for given S, T there are $M \in \mathcal{MO}(k)$ and $v \in \mathbb{R}^k$ such that $\delta_p^C(S,T) = \delta_p(S,\, M \cdot T + v)$.

Lemma 7. *Definition 7 is well-defined.*

Proof. It is necessary to show that $M^* \in \mathcal{MO}(k)$ and $v^* \in \mathbb{R}^k$ exist such that

$$\delta_p^C(S,T) \;=\; \delta_p\left(S,\, M^* \cdot T + v^*\right). \tag{4.1}$$

To that, the range of parameters is restricted to a bounded set first. Then, the Bolzano-Weierstrass theorem[1] is used to show that such M^* and v^* exist.

Now, consider arbitrary but fixed $S, T \in \mathcal{T}$, $M \in \mathcal{MO}(k)$, and $v \in \mathbb{R}^k$. The static time series consisting only of v and $0 \in \mathbb{R}^k$ are denoted by $\bar{v} \coloneqq (v, \cdots, v) \in \mathcal{T}$ and $\bar{0} = (0, \cdots, 0) \in \mathcal{T}$ with $\#\bar{v} = \#0 = \#S$, respectively.

At first, a finite boundary for the vector v^* in Equation 4.1 is necessary. To that, the following inequation is crucial since it finds an estimation that is independent of the transformation matrix M. To simplify the calculation, a first step is to exploit the equivalence of all norms in finite dimensional vector spaces, i. e., for fixed $\#S < \infty$, a constant $C > 0$ exists with $\|\cdot\|_p \geqslant C \cdot \|\cdot\|_1$; and therefore, $\delta_p \geqslant C \cdot \delta_1$. Furthermore, it uses the definition of δ_1 and the axioms of a norm[2]:

$$\begin{aligned}
\delta_p(S,\; M \cdot T + v) \;&\geqslant\; C \cdot \delta_1(S,\; M \cdot T + v) \\
&\geqslant\; C \cdot (\delta_1(M \cdot T + v,\; \bar{0}) - \delta_1(S, \bar{0})) \\
&\geqslant\; C \cdot (\delta_1(\bar{v},\; \bar{0}) - \delta_1(M \cdot T,\; \bar{0}) - \delta_1(S, \bar{0})) \\
&=\; C \cdot (\delta_1(\bar{v}, \bar{0}) - \delta_1(T, \bar{0}) - \delta_1(S, \bar{0}))
\end{aligned} \tag{4.2}$$

Since $\delta_1(\bar{v}, \bar{0}) = \|\mathtt{d}_2(\bar{v}, \bar{0})\|_1 = n \cdot \|v\|_2$, Inequation 4.2 yields

$$\delta_p(S,\; M \cdot T + v) \;\geqslant\; C \cdot (n \cdot \|v\|_2 \;-\; \delta_1(T, \bar{0}) - \delta_1(S, \bar{0})). \tag{4.3}$$

Now, consider r as boundary with

$$r \;\coloneqq\; \frac{1}{C} \cdot \frac{\delta_1(S,T) + C \cdot \delta_1(T, \bar{0}) + C \cdot \delta_1(S, \bar{0})}{n}.$$

For $v \in \mathbb{R}^k$ with $\|v\|_2 > r$, substituting $\|v\|_2$ in Inequation 4.3 yields

$$\delta_p(S,\; M \cdot T + v) > \delta_p(S,T) \;\geqslant\; \inf_{M' \in \mathcal{MO}(k),\, v' \in \mathbb{R}^k} \delta_p\left(S,\, M' \cdot T + v'\right)$$

for every $M \in \mathcal{MO}(k)$.[3] Hence, it suffices to restrict attention to $v' \in \mathbb{R}^k$ with $\|v'\|_2 \leqslant r$, i. e.,

[1] The Bolzano-Weierstrass theorem claims that each bounded sequence of elements in a metric space contains a convergent subsequence.

[2] This calculation uses the trick $\|A\| = \|A - B + B\| \leqslant \|A - B\| + \|B\| \iff \|A\| - \|B\| \leqslant \|A - B\|$ for $A, B \in \mathbb{R}^k$.

[3] For the latter inequality, consider $v' = 0 \in \mathbb{R}^k$ and M' the identity matrix.

$$\delta_p^C(S,T) \;=\; \inf_{M' \in \mathcal{MO}(k),\, v' \in V} \delta_p\left(S,\; M' \cdot T + v'\right),$$

for $V \coloneqq \left\{v \in \mathbb{R}^k : \|v\|_2 \leqslant r\right\}$.

To apply the Bolzano-Weierstraß theorem, the distance function is reinterpreted as follows: Let $X \coloneqq \mathcal{MO}(k) \times V$, let $Y \coloneqq \mathbb{R}_{\geqslant 0}$, and let $f : X \longrightarrow Y$ be defined via

$$f(M,v) \;\coloneqq\; \delta_p(S,\; M \cdot T + v).$$

Furthermore, consider each element $(M,v) \in X$ as a vector in $\mathbb{R}^{k^2+k}$ and choose the Euclidean distance d_2 as a metric on X. It is straightforward to see that, with respect to this metric,

(1) X is a bounded set, and

(2) f is a continuous function.

Now, consider an arbitrary sequence $\xi \coloneqq (M_i, v_i)_{i \in \mathbb{N}}$ with $M_i \in \mathcal{MO}(k)$ and $v_i \in V$ for all $i \in \mathbb{N}$ such that

$$\delta_p^C(S,T) \;=\; \lim_{i \to \infty} \delta_p(S,\; M_i \cdot T + v_i) \;\;=\;\; \lim_{i \to \infty} f(M_i, v_i).$$

So far, it is not clear that the sequence of parameters ξ converges even though the sequence of values $f(M_i, v_i)$ converges. However, since $X = \mathcal{MO}(k) \times V$ is a bounded set (w.r.t. d_2) the sequence ξ is bounded. Therefore, the Bolzano-Weierstrass theorem is applicable and yields a convergent subsequence $\xi' = (M_{i_j}, v_{i_j})_{j \in \mathbb{N}}$ with $i_1 < i_2 < \cdots$.

In the image space, both sequences $f(\xi)$ and $f(\xi')$ converge against the same value since ξ' is a subsequence of ξ, i. e., $\lim_{j \to \infty} f(M_{i_j}, v_{i_j}) = \lim_{i \to \infty} f(M_i, v_i)$. In the parameter space, let $(M^*, v^*) \in X$ be the limit of the sequence ξ', i. e.,

$$(M^*, v^*) \;\coloneqq\; \lim_{j \to \infty} (M_{i_j}, v_{i_j}).$$

Considering that f is a continuous function, the following equation holds:

$$\begin{aligned} \delta_p^C \;&=\; \lim_{i \to \infty} f(M_i, v_i) \\ &=\; \lim_{j \to \infty} f(M_{i_j}, v_{i_j}) \\ &=\; f(M^*, v^*) \\ &=\; \delta_p(S,\; M^* \cdot T + v^*) \end{aligned}$$

Therefore, Definition 7 is well-defined. □

Lemma 8. The congruence distance δ_p^C is invariant under rotation, translation, and reflection.

Proof. Let $S, T \in \mathcal{T}$ be two time series with $n = \#S = \#T$. For $M \in \mathcal{MO}(k)$, the equation

$$\begin{aligned}
\delta_p(S,T) &\coloneqq \| \, \mathtt{d}_2(S,T) \, \|_p \\
&\coloneqq \left\| \, (\| \, s_i - t_i \, \|_2 \,)_{0 \leqslant i < n} \, \right\|_p \\
&= \left\| \, (\| \, M(s_i - t_i) \, \|_2 \,)_{0 \leqslant i < n} \, \right\|_p \\
&= \left\| \, (\| \, (M \cdot s_i + v) \; - \; (M \cdot t_i + v)) \, \|_2)_{0 \leqslant i < n} \, \right\|_p \\
&= \delta_p(M \cdot S + v, \; M \cdot T + v) \qquad (4.4)
\end{aligned}$$

holds. Therefore, δ_p^C is invariant under isometric transformations when both time series are transformed equally.

Using Equation (4.4) and considering $M^* \in \mathcal{MO}(k)$ and $v^* \in \mathbb{R}^k$ such that $\delta_p^C(S,T) = \delta_p(S, M^* \cdot T + v^*)$, which exist according to Lemma 7, the inequation

$$\begin{aligned}
\delta_p^C(S,T)\| &= \delta_p \left(\; S, \; M^* \cdot T + v^* \; \right) \\
&= \delta_p \left(\; M \cdot S + v, \; M \cdot M^* \cdot T \; + \; (M \cdot v^* + v) \; \right) \\
&\geqslant \inf_{M' \in \mathcal{MO}(k), v' \in \mathbb{R}^k} \delta_p \left(\; M \cdot S + v, \; M' \cdot T + v' \; \right) \\
&= \delta_p^C(M \cdot S + v, T) \qquad (4.5)
\end{aligned}$$

holds. For $M^* \in \mathcal{MO}(k)$ and $v^* \in \mathbb{R}^k$ with $\delta_p^C(M \cdot S + v, T) = \delta_p(M \cdot S + v, \; M^* \cdot T + v^*)$,

$$\begin{aligned}
\delta_p^C(M \cdot S + v, T)\| &\coloneqq \inf_{M' \in \mathcal{MO}(k), v' \in \mathbb{R}^k} \delta_p \left(\; M \cdot S + v, \; M' \cdot T + v' \; \right) \\
&= \delta_p \left(\; M \cdot S + v, \; M^* \cdot T + v^* \; \right) \\
&= \delta_p \left(\; S, \; M^{-1} \cdot M^* \cdot T \; + \; \left(M^{-1} \cdot v^* - v \right) \; \right) \\
&\geqslant \inf_{M' \in \mathcal{MO}(k), v' \in \mathbb{R}^k} \delta_p(S, \; M' \cdot T + v') \\
&= \delta_p^C(S,T) \qquad (4.6)
\end{aligned}$$

holds analogously. Equation (4.5) and (4.6) yield $\delta_p^C(S,T) = \delta_p^C(M \cdot S + v, T)$ for every $M \in \mathcal{MO}(k)$ and $v \in \mathbb{R}^k$, i. e., δ_p^C is invariant under isometric transformations on the first argument.[4]

[4]Note that multiplication with an arbitrary matrix $M \in \mathcal{MO}(k)$ and adding an arbitrary vector $v \in \mathbb{R}^k$ represents all isometric transformations already.

For arbitrary $M_1, M_2 \in \mathcal{MO}(k)$ and $v_1, v_2 \in \mathbb{R}^k$, both results prove the claim of the lemma when considering inverse transformations:

$$\begin{aligned}
\delta_p^C(S,T) &= \delta_p^C\,(\; M_2 \cdot S + v_2,\; M_2 \cdot T + v_2\;) \\
&= \delta_p^C\,\big(\; M_2^{-1}(M_2 \cdot S + v_2) - M_2^{-1} \cdot v_2,\; M_2 \cdot T + v_2\;\big) \\
&= \delta_p^C\,(\; S,\; M_2 \cdot T + v_2\;) \\
&= \delta_p^C\,(\; M_1 \cdot S + v_1,\; M_2 \cdot T + v_2\;)
\end{aligned}$$

□

Utilizing Lemma 8, the following proposition shows that the congruence distance δ_p^C yields a pseudo metric space. Although this proposition is not used further on, it is an interesting result relevant to the topic of this thesis.

Proposition 1. *$(\mathcal{T}_n, \delta_p^C)$ is a pseudo metric space.*

Proof. It is easy to see that δ_p and thus δ_p^C are symmetric functions, i.e., $\delta_p^C(S,T) = \delta_p^C(T,S)$. Furthermore, the triangle inequality for δ_p follows from the axioms of the norm.

To prove the triangle inequality of δ_p^C, let $M_1, M_2 \in \mathcal{MO}(k)$ and $v_1, v_2 \in \mathbb{R}^k$ such that $\delta_p^C(S,T) = \delta_p(M_1 \cdot S + v_1, T)$ and $\delta_p^C(T,U) = \delta_p(T, M_2 \cdot U + v_2)$. These vectors and matrices exist according to Lemma 7 and Lemma 8. Then, the triangle inequality follows:

$$\begin{aligned}
\delta_p^C(S,U) &= \delta_p^C(M_1 \cdot S + v_1,\; M_2 \cdot U + v_2) \\
&\leqslant \delta_p(M_1 \cdot S + v_1,\; M_2 \cdot U + v_2) \\
&\leqslant \delta_p(M_1 \cdot S + v_1,\; T) + \delta_p(T,\; M_2 \cdot U + v_2) \\
&= \delta_p^C(S,T) + \delta_p^C(T,U).
\end{aligned}$$

□

Calculating $\delta_p^C(S,T)$ for arbitrary $S, T \in \mathcal{T}$ is a nonlinear optimization problem that can be solved using numeric solvers. Unfortunately, the problem is computationally difficult: The next subsection shows that the calculation of δ_1^C is NP-hard already.

4.2.2 NP-Hardness

This section restricts attention to $\delta_1 = \|\mathtt{d}_2\|_1$ and the according congruence distance δ_1^C. Consider the following problem:

δ_1^C-Computation

Input: A number $k \in \mathbb{N}$ and two time series S and T in $\mathbb{R}^k$ of equal length.

Task: Compute (a suitable representation of) the number $\delta_1^C(S,T)$.

This section's main result is:

Theorem 3. If P $\neq$ NP then δ_1^C-Computation cannot be solved in polynomial time.

The remainder of this section is devoted to the proof of Theorem 3, which constructs a reduction from the NP-complete problem 1-in-3-Sat. Recall that 1-in-3-Sat is the problem where the input consists of a propositional formula Φ in 3-cnf, i.e., in conjunctive normal form where each clause is a disjunction of literals over three distinct variables. The task is to decide whether there is an assignment α that maps the variables occurring in Φ to the truth values 0 or 1 such that in each disjunctive clause of Φ exactly one literal is satisfied by α; such an assignment α will be called a *1-in-3 model of* Φ.

Outline: The reduction from 1-in-3-Sat to the δ_1^C-Computation will proceed as follows: A given 3-cnf formula Φ with k variables will be mapped to two time series $\bar{S}_\Phi$ and $\bar{T}_\Phi$ over $\mathbb{R}^k$ that represent the formula Φ and its variables, respectively. The basic idea for the choice of $\bar{S}_\Phi$ and $\bar{T}_\Phi$ is that each dimension of $\mathbb{R}^k$ represents exactly one of the variables. An orthogonal matrix mirroring the i-th dimension will then correspond to negating the i-th variable. The construction of $\bar{S}_\Phi$ and $\bar{T}_\Phi$ will ensure that the following is true for a certain number $c(\Phi)$: $\delta_1^C(\bar{S}_\Phi, \bar{T}_\Phi) = c(\Phi) \iff$ there is a 1-in-3 model of Φ.

Notation: The following notation will be convenient to formulate the proof. For a propositional formula Φ with k variables, $V_1, \ldots, V_k$ denote the variables occurring in Φ. A *literal* over a variable V_i is a formula $L_i \in \{V_i, \neg V_i\}$. A *disjunctive (conjunctive) 3-clause* is a formula $\Psi_I = \bigvee_{i\in I} L_i$ ($\Psi_I = \bigwedge_{i\in I} L_i$) with $L_i \in \{V_i, \neg V_i\}$, $\#I = 3$, and $I \subseteq [1,k]$[5]. A *3-cnf formula* is a formula $\Phi = \bigwedge_{j=1}^m \Psi_j$ where $m \geqslant 1$ and each Ψ_j is a disjunctive 3-clause.

Furthermore, the following notation is used for concatenating time series. Let $\ell \geqslant 1$, $n_j \in \mathbb{N}$ for $1 \leqslant j \leqslant \ell$, and let $S_j = (s_0^j, \cdots, s_{n_j-1}^j)$ be a time series over $\mathbb{R}^k$ for each $j \in [1,\ell]$. Then, their concatenation is denoted as

$$S_1 \times \cdots \times S_\ell \;:=\; \big(\, s_1^1, \cdots, s_{n_1}^1,\; \cdots,\; s_1^\ell, \cdots, s_{n_\ell}^\ell \,\big).$$

If $i_1 < \cdots < i_\ell$ is an increasing sequence of integers and S_{i_j} is a time series over $\mathbb{R}^k$ for each $j \in [1,\ell]$ then for $I := \{i_1, \cdots, i_\ell\}$ the concatenation is abbreviated with

$$\bigotimes_{i\in I} S_i \;:=\; S_{i_1} \times \cdots \times S_{i_\ell}.$$

[5]Except the proof of Theorem 3, this section assumes no trivial 3-clauses, e. g., $\Psi = V_i \wedge \neg V_i$ and $\Psi = V_i \vee V_i$ are excluded.

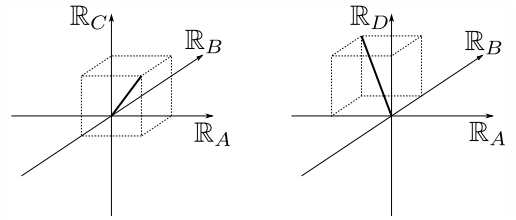


Figure 4.1: A mapping of the two formulas $A \wedge \neg B \wedge C$ (left) and $\neg A \wedge B \wedge D$ (right) to $\mathbb{R}^k$

From a 3-cnf formula Φ to the time series $\bar{S}_\Phi$ and $\bar{T}_\Phi$: Let k be the number of variables occurring in a given 3-cnf formula $\Phi = \bigwedge_{j=1}^{m} \Psi_j$ where $m \geqslant 1$ and each $\Psi_j = \bigvee_{i \in I_j} L_i$ is a disjunctive 3-clause with $L_i \in \{V_i, \neg V_i\}$, $\#I_j = 3$, and $I_j \subseteq [1, k]$.

For a disjunctive 3-clause $\Psi = \bigvee_{j \in I} L_j$, let

$$\Psi' \;\coloneqq\; \bigvee_{j \in I} \Big(\underbrace{L_j \;\wedge \bigwedge_{i \in I \setminus \{j\}} \neg L_i}_{=:\; \Gamma_j} \Big).$$

Clearly, an assignment α satisfies Ψ' iff it is a 1-in-3 model of Ψ; and α satisfies $\Phi' \coloneqq \bigwedge_{j=1}^{m} \Psi'_j$ iff it is a 1-in-3 model of $\Phi = \bigwedge_{j=1}^{m} \Psi_j$. The formulas Γ_j for $j \in I$ are called *the conjunctive 3-clauses implicit in* Ψ.

Regarding formulas as purely syntactical constructs, an embedding of formulas into $\mathbb{R}^k$ can be defined recursively. To that, consider the following embedding θ of variables, literals, and conjunctive 3-clauses into $\mathbb{R}^k$: For each $i \in [1, k]$ let

$$\theta(V_i) \coloneqq e_i.$$

For each formula Φ with (at most) k variables, let

$$\theta(\neg\Phi) \coloneqq -\theta(\Phi).$$

For a conjunction $\Gamma = \bigwedge_{i \in I} L_i$, let

$$\theta(\Gamma) \;\coloneqq\; \sum_{i \in I} \theta(L_i).$$

Figure 4.1 sketches some examples of this mapping. In particular, for an *implicit conjunctive 3-clause* Γ_j as defined above, the mapping yields

$$\theta(\Gamma_j) \;= \theta\left(L_j \wedge \bigwedge_{i \in I \setminus \{j\}} \neg L_i \right) \;\coloneqq\; \theta\left(L_j\right) - \sum_{i \in I \setminus \{j\}} \theta(L_i).$$

For each index set $I \subseteq [1,k]$, let

$$e_I \;\coloneqq\; \sum_{i\in I} e_i$$

and for each disjunctive 3-clause $\Psi = \bigvee_{i\in I} L_i$ define the following time series over $\mathbb{R}^k$:

$$\begin{aligned}
S'_\Psi \;&\coloneqq \bigotimes_{i\in I} (6e_i, -6e_i) \qquad , \qquad T'_\Psi \;\coloneqq \bigotimes_{i\in I} (6e_i, 6e_i),\\
S_\Psi \;&\coloneqq \bigotimes_{j\in I} (\gamma_j) \qquad , \qquad T_\Psi \;\coloneqq (\, e_I,\; e_I,\; e_I\,),\\
\tilde{S}_\Psi \;&\coloneqq S'_\Psi \times S_\Psi \qquad , \qquad \tilde{T}_\Psi \;\coloneqq T'_\Psi \times T_\Psi.
\end{aligned} \tag{4.7}$$

To anticipate an understanding of the choice of these time series, figure that $\delta_1^C(S'_\Psi, T'_\Psi)$ will be minimized with a transformation that rotates around each axis by either 0 degrees or 180 degrees, i. e., these sub time series force a minimizing transformation matrix to enter binary states regarding the transformation of each e_i, respectively. On the other hand, minimizing the congruence distance of S_Ψ and T_Ψ corresponds to setting the propositional variables such that the 3-clause Ψ is fulfilled.

For a 3-cnf formula $\Phi = \bigwedge_{j=1}^m \Psi_j$, all these time series will be concatened to the two time series

$$S_\Phi \;\coloneqq \bigotimes_{j=1}^{m} \left(\tilde{S}_{\Psi_j}\right) \qquad , \qquad T_\Phi \;\coloneqq \bigotimes_{j=1}^{m} \left(\tilde{T}_{\Psi_j}\right).$$

In the end, the time series are concatenated with their mirrored duplicates:[6]

$$\bar{S}_\Phi \coloneqq\; S_\Phi \times -S_\Phi \qquad , \qquad \bar{T}_\Phi \coloneqq\; T_\Phi \times -T_\Phi.$$

The aim is to compute a number $c(\Phi)$ such that the following is true: $\delta_1^C(\bar{S}_\Phi, \bar{T}_\Phi) = c(\Phi)$ iff Φ has a 1-in-3 model. Several steps are necessary to obtain this equivalence, the first of which is to compute a number $c^{\mathcal{MO}}(\Phi)$ such that Φ has a 1-in-3 model iff $\delta_1^{\mathcal{MO}}(S_\Phi, T_\Phi) = c^{\mathcal{MO}}(\Phi)$ where

$$\delta_1^{\mathcal{MO}}(S,T) \;\coloneqq \min_{M\in\mathcal{MO}(k)} \delta_1(S,\; M\cdot T), \tag{4.8}$$

i. e., Φ has a 1-in-3-model iff S_Φ and T_Φ are congruent with distance $c^{\mathcal{MO}}(\Phi)$ while forcing the translation to be 0.

[6] For the congruence distance, this will force the translation vector of the minimizing transformation to be 0.

From $\mathcal{MO}(k)$ to propositional variable assignments: This section's Lemmas, Theorems, and Corollaries use the following definitions that relate orthogonal matrices with propositional variable assignments.

Definition 8. A matrix $M \in \mathcal{MO}(k)$ is called a Boolean matrix iff

$$\forall 1 \leqslant i \leqslant k : M \cdot e_i \in \{e_i, -e_i\}.$$

The set of all Boolean matrices is denoted with $\mathcal{O}(k)_{\mathcal{L}}$.

Definition 9. For each Boolean matrix $M_B \in \mathcal{O}(k)_{\mathcal{L}}$, let $\eta(M_B)$ be the assignment α with

$$\alpha(V_i) = \begin{cases} 1 & \text{if } m_{i,i} = 1; \\ -1 & \text{else.} \end{cases}$$

For each orthogonal matrix $M \in \mathcal{MO}(k)$, let $\hat{\eta}(M)$ be the assignment α with

$$\alpha(V_i) = \begin{cases} 1 & \text{if } m_{i,i} \geqslant 0; \\ -1 & \text{else.} \end{cases}$$

Clearly, η is a bijection between the Boolean matrices $\mathcal{O}(k)_{\mathcal{L}}$ and the set of all assignments to the propositional variables $V_1, \ldots, V_k$. By definition, $\hat{\eta}(M_B) = \eta(M_B)$ holds for each Boolean matrix $M_B \in \mathcal{O}(k)_{\mathcal{L}}$, i. e., $\hat{\eta}$ is an extension of η to all orthogonal matrices. Therefore, $\hat{\eta}$ partitions $\mathcal{MO}(k)$, i. e.,

$$\mathcal{MO}(k) = \bigcup_{M_B \in \mathcal{O}(k)_{\mathcal{L}}} \hat{\eta}^{-1}(\eta(M_B))$$

is disjunctive. This partition defines an equivalence relation via

$$M \equiv M' \ :\Longleftrightarrow\ \hat{\eta}(M) = \hat{\eta}(M')$$

for all orthogonal matrices $M, M' \in \mathcal{MO}(k)$. Furthermore,

$$M \equiv M' \implies \hat{\eta}(M) \models \Phi \Longleftrightarrow \hat{\eta}(M') \models \Phi$$

holds by definition for arbitrary propositional formulas Φ.

Relating $\delta_1^{\mathcal{MO}}(S_\Phi, T_\Phi)$ with 1-in-3 models of Φ: The next lemma lifts Thales' Theorem to multidimensional Vector Spaces, which will help prove Theorem 3.

Lemma 9. *Let $M \in \mathcal{MO}(k)$, $i \in [1, k]$, $a_i := \mathtt{d}_2(e_i, Me_i)$, and $b_i := \mathtt{d}_2(-e_i, Me_i)$. Then,*

$$b_i = \sqrt{4 - a_i^2} \qquad \textit{and} \qquad a_i = \sqrt{4 - b_i^2}.$$

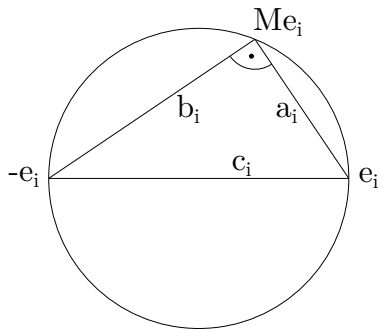


Figure 4.2: Thales' theorem sketched in $\mathbb{R}^k$, where $e_i \in \mathbb{R}^k$ and $M \in \mathcal{MO}(k)$. The side c_i is a diameter; a_i and b_i connect both endings to a third point (Me_i) on the circle.

Figure 4.2 sketches the situation described by Lemma 9.

Proof. Let $c_i := \mathsf{d}_2(e_i, -e_i)$. For the special case where $k = 2$, Thales' Theorem claims that $a_i^2 + b_i^2 = c_i^2$. The same holds for arbitrary k, as the following computation shows.

Clearly, $c_i = \mathsf{d}_2(e_i, -e_i) = \|2e_i\|_2 = 2$, and thus $c_i^2 = 4$. Furthermore,

$$\begin{aligned} a_i^2 &= \mathsf{d}_2(e_i, Me_i)^2 \\ &= \|e_i - Me_i\|_2^2 \\ &= \langle e_i - Me_i, e_i - Me_i\rangle \\ &= \langle e_i, e_i\rangle + \langle Me_i, Me_i\rangle - 2\langle e_i, Me_i\rangle \\ &= 2 - 2\langle e_i, Me_i\rangle \end{aligned}$$

and

$$\begin{aligned} b_i^2 &= \mathsf{d}_2(-e_i, Me_i)^2 \\ &= \|-e_i - Me_i\|_2^2 \\ &= \|e_i + Me_i\|_2^2 \\ &= \langle e_i + Me_i, e_i + Me_i\rangle \\ &= \langle e_i, e_i\rangle + \langle Me_i, Me_i\rangle + 2\langle e_i, Me_i\rangle \\ &= 2 + 2\langle e_i, Me_i\rangle. \end{aligned}$$

Thus,

$$a_i^2 + b_i^2 \;=\; 2 - 2\langle e_i, Me_i\rangle + 2 + 2\langle e_i, Me_i\rangle \;=\; 4 \;=\; c_i^2,$$

and therefore $b_i = \sqrt{4 - a_i^2}$ and $a_i = \sqrt{4 - b_i^2}$. □

For a disjunctive 3-clause Ψ and a matrix $M \in \mathcal{MO}(k)$, let

$$\begin{aligned}\delta_\Psi(M) &\coloneqq \delta_1(\tilde{S}_\Psi,\ M \cdot \tilde{T}_\Psi) \\ &= \delta_1(S'_\Psi,\ M \cdot T'_\Psi)\ +\ \delta_1(S_\Psi,\ M \cdot T_\Psi).\end{aligned}$$

The next lemmas will gather information on the size of $\delta_\Psi(M)$.

Lemma 10. Let $\Psi = \bigvee_{i\in I} L_i$ be a disjunctive 3-clause, let $M \in \mathcal{MO}(k)$. Then,

$$\delta_1(S'_\Psi,\ M \cdot T'_\Psi)\ =\ 6 \cdot \sum_{i\in I} \big(\mathtt{d}_2(e_i, Me_i) + \mathtt{d}_2(-e_i, Me_i)\big).$$

Proof. Recalling that $\delta_1 = \|\mathtt{d}_2\|_1$, the lemma holds trivially for the particular choice of S'_Ψ and T'_Ψ. □

Lemma 11. Let $\Psi = \bigvee_{i\in I} L_i$ be a disjunctive 3-clause.

(a) For each orthogonal matrix $M \in \mathcal{MO}(k)$, the inequation

$$\begin{aligned}\delta_1(S_\Psi,\ M \cdot T_\Psi)\ \geqslant\ & B_\Psi(M) \\ & -\ 3 \cdot \sum_{i\in I} \min(\mathtt{d}_2(e_i, Me_i), \mathtt{d}_2(-e_i, Me_i)) \qquad (4.9)\end{aligned}$$

holds where $B_\Psi(M) = 4\sqrt{2}$ if $\hat{\eta}(M) \models \Gamma$ for some conjunctive 3-clause Γ implicit in Ψ and $B_\Psi(M) \in \{6, 4+2\sqrt{3}, 6\sqrt{2}\}$ else.

(b) For each Boolean matrix $M_B \in \mathcal{O}(k)_\mathcal{L}$,

$$\delta_1(S_\Psi,\ M_B \cdot T_\Psi)\ =\ B_\Psi(M_B)$$

holds where $B_\Psi(M_B)$ is as in (a).

Proof. Let Γ_j, for $j \in I$, be the conjunctive 3-clauses implicit in Ψ, and let $\gamma_j = \theta(\Gamma_j)$.

For proving (a), let $M \in \mathcal{MO}(k)$ be an arbitrary orthogonal matrix. By definition of S_Ψ and T_Ψ, the following holds:

$$\delta_1(S_\Psi,\ M \cdot T_\Psi)\ =\ \sum_{j\in I} \mathtt{d}_2(\gamma_j,\ Me_I). \qquad (4.10)$$

The triangle inequality and the symmetry yield $\mathtt{d}(x,z) \geqslant \mathtt{d}(x,y) - \mathtt{d}(y,z)$ for all pseudo metric spaces $(\mathbb{M}, \mathtt{d})$ and all $x, y, z \in \mathbb{M}$. Thus, for any vector $v \in \mathbb{R}^k$ and for any $j \in I$ the inequation

$$\mathtt{d}_2(\gamma_j, Me_I)\ \geqslant\ \mathtt{d}_2(\gamma_j, v)\ -\ \mathtt{d}_2(v, Me_I),$$

holds, and hence

$$\begin{aligned}\delta_1(S_\Psi, M \cdot T_\Psi) &\geqslant \sum_{j\in I} \Big(\mathtt{d}_2(\gamma_j,\ v) - \mathtt{d}_2(v, Me_I) \Big) \\ &= \left(\sum_{j\in I} \mathtt{d}_2(\gamma_j, v) \right) - 3 \cdot \mathtt{d}_2(v, Me_I). \end{aligned} \tag{4.11}$$

Choose $v \in \mathbb{R}^k$ as follows: Let $v \coloneqq \sum_{i\in I} s_i e_i$ where $s_i \coloneqq 1$ if $\mathtt{d}_2(e_i, Me_i) \leqslant \mathtt{d}_2(-e_i, Me_i)$, and $s_i \coloneqq -1$ otherwise. Then,

$$\begin{aligned}\mathtt{d}_2(v, Me_I) &= \|v - Me_I\|_2 \\ &= \left\| \sum_{i\in I} (s_i e_i - Me_i) \right\|_2 \\ &\leqslant \sum_{i\in I} \|s_i e_i - Me_i\|_2 \\ &= \sum_{i\in I} \mathtt{d}_2(s_i e_i, Me_i). \end{aligned}$$

Note that $\mathtt{d}_2(s_i e_i,\ Me_i)$ is equal to a_i if $s_i = 1$, and it is equal to $\mathtt{d}_2(-e_i, Me_i)$ if $s_i = -1$ (cf. Figure 4.2). Thus, due to the choice of s_i, the equation $\mathtt{d}_2(s_i e_i, Me_i) = \min(\mathtt{d}_2(e_i, Me_i), \mathtt{d}_2(-e_i, Me_i))$ holds, and hence Inequation 4.11 continues with

$$\begin{aligned}\delta_1(S_\Psi, M \cdot T_\Psi) \ \geqslant\ & \left(\sum_{j\in I} \mathtt{d}_2(\gamma_j, v) \right) \\ & - \ 3 \cdot \sum_{i\in I} \min \Big(\ \mathtt{d}_2(e_i, Me_i),\ \mathtt{d}_2(-e_i, Me_i) \ \Big). \end{aligned} \tag{4.12}$$

To reach Inequation (4.9) requires to show that $\sum_{j\in I} \mathtt{d}_2(\gamma_j, v) \geqslant B_\Psi(M)$. For simplicity, consider w. l. o. g. the case where $I = \{1, 2, 3\}$. Let $l_i = \theta(L_i)$ for $i \in I$; thus, $l_i \in \{e_i, -e_i\}$. Then, w. l. o. g.

$$\begin{aligned}\gamma_1 &= l_1 - l_2 - l_3, \\ \gamma_2 &= -l_1 + l_2 - l_3, \quad \text{and} \\ \gamma_3 &= -l_1 - l_3 + l_3. \end{aligned}$$

The following case distinction according to v shows that $\sum_{j\in I} \mathtt{d}_2(\gamma_j, v) \geqslant B_\Psi(M)$.

Case 1: $v = \gamma_i$ *for some* $i \in I$, i. e., $\hat{\eta}(M) \models \Gamma_i$ by definition of v. In this case, $\mathtt{d}_2(\gamma_i, v) = 0$. Then, $\mathtt{d}_2(\gamma_j, v)$ resolves to $\sqrt{8} = 2\sqrt{2}$ for each $j \in I \setminus \{i\}$ by substituting the values from above. Thus, $\sum_{j\in I} \mathtt{d}_2(\gamma_j, v) = \mathtt{d}_2(\gamma_1, v) + \mathtt{d}_2(\gamma_2, v) + \mathtt{d}_2(\gamma_3, v) = 4\sqrt{2} \approx 5.66$.

Case 2: $v = -l_1 - l_2 - l_3$. Then, the equation $\mathtt{d}_2(\gamma_j, v) = \sqrt{4} = 2$ holds for each $j \in I$; thus, $\sum_{j\in I} \mathtt{d}_2(\gamma_j, v) = 3 \cdot 2 = 6$.

Case 3: $v = -\gamma_i$ *for some* $i \in I$. In this case, $\mathtt{d}_2(\gamma_i, v) = 2 \cdot \|\gamma_i\|_2 = 2\sqrt{3}$. Furthermore, $\mathtt{d}_2(\gamma_j, v)$ resolves to $\sqrt{4} = 2$ for each $j \in I \setminus \{i\}$; thus, $\sum_{j\in I} \mathtt{d}_2(\gamma_j, v) = 4 + 2\sqrt{3} \approx 7.46$.

Case 4: $v = l_1 + l_2 + l_3$. Then, for each $j \in I$ the equation $\mathtt{d}_2(\gamma_j, v) = \sqrt{4+4} = \sqrt{8} = 2\sqrt{2}$ holds; thus, $\sum_{j\in I} \mathtt{d}_2(\gamma_j, v) = 3 \cdot 2\sqrt{2} = 6\sqrt{2} \approx 8.49$.

Note that Cases 1–4 comprise all possible cases for v and in all these cases $\sum_{j\in I} \mathtt{d}_2(\gamma_j, v) \geqslant B_\Psi(M)$. Especially, $\sum_{j\in I} \mathtt{d}_2(\gamma_j, v) \geqslant 4\sqrt{2}$ if there is a conjunctive 3-clause Γ_j implicit in Ψ such that $\hat{\eta}(M) \models \Gamma_i$ (Case 1) and $\sum_{j\in I} \mathtt{d}_2(\gamma_j, v) \geqslant 6$ else (Cases 2–4). Together with (4.12), this yields that Inequation (4.9) is correct, which completes the proof of (a).

Now, for a Boolean matrix $M \in \mathcal{O}(k)_\mathcal{L}$, $M \cdot e_I = v$ holds by definition of v. With this matrix and the case distinction from above, Equation (4.10) resolves to

$$\delta_1(S_\Psi,\ M \cdot T_\Psi) \quad = \quad \sum_{j\in I} \mathtt{d}_2(\gamma_j, v) \quad = \quad B_\Psi(M),$$

which completes the proof of (b). □

The following corollary utilizes the previous two lemmas.

Corollary 1. Let Ψ be a disjunctive 3-clause. With $B_\Psi(M)$ as in Lemma 11, the inequation

$$\delta_\Psi(M) \quad \geqslant \quad 36 \ + \ B_\Psi(M) \tag{4.13}$$

holds for arbitrary orthogonal matrices $M \in \mathcal{MO}(k)$. For Boolean matrices $M \in \mathcal{O}(k)_\mathcal{L}$, the equality holds in Inequation (4.13).

Proof. For $M \in \mathcal{MO}(k)$, Lemmas 10 and 11 yield

$$\begin{aligned}
\delta_\Psi(M) &= \delta_1(S'_\Psi, M \cdot T'_\psi) + \delta_1(S_\Psi, M \cdot T_\Psi) \\
&\geqslant 6 \cdot \sum_{i \in I} (\mathtt{d}_2(e_i, Me_i) + \mathtt{d}_2(-e_i, Me_i)) \\
&\quad + B_\Psi(M) - 3 \cdot \sum_{i \in I} \min(\mathtt{d}_2(e_i, Me_i), \mathtt{d}_2(-e_i, Me_i)) \\
&= B_\Psi(M) + 3 \cdot \sum_{i \in I} (\mathtt{d}_2(e_i, Me_i) + \mathtt{d}_2(-e_i, Me_i)) \\
&\quad + 3 \cdot \sum_{i \in I} \max(\mathtt{d}_2(e_i, Me_i), \mathtt{d}_2(-e_i, Me_i)) \\
&\geqslant B_\Psi(M) + 3 \cdot \sum_{i \in I} (\mathtt{d}_2(e_i, M \cdot e_i), \ \mathtt{d}_2(M \cdot e_i, -e_i)) \\
&\geqslant B_\Psi(M) + 3 \cdot \sum_{i \in I} \mathtt{d}_2(e_i, -e_2) \\
&= 36 + \delta_\Psi(M),
\end{aligned}$$

which proves the first part of the corollary.

For the second part of the corollary, let $M_B \in \mathcal{O}(k)_\mathcal{L}$ be an arbitrary Boolean matrix. Lemmas 10 and 11 yield

$$\begin{aligned}
\delta_\Psi(M_B) &= \delta_1(S'_\Psi, \ M_B \cdot T'_\psi) + \delta_1(S_\Psi, \ M_B \cdot T_\Psi) \\
&= 6 \cdot \sum_{i \in I} \Big(\mathtt{d}_2(e_i, M_B e_i) + \mathtt{d}_2(-e_i, M_B e_i) \Big) + B_\Psi(M_B).
\end{aligned}$$

Since $M_B \cdot e_i \in \{e_i, -e_i\}$, the following equation holds:

$$\mathtt{d}_2(e_i, M_B \cdot e_i) + \mathtt{d}_2(-e_i, M_B \cdot e_i) = \mathtt{d}_2(e_i, -e_i) = 2$$

Hence, with $\#I = 3$,

$$6 \cdot \sum_{i \in I} \Big(\mathtt{d}_2(e_i, M_B \cdot e_i) + \mathtt{d}_2(-e_i, M_B \cdot e_i) \Big) = 6 \cdot 3 \cdot 2 = 36,$$

and thus

$$\delta_\Psi(M_B) = 36 + B_\Psi(M_B),$$

which proves the second part of the corollary. □

Lemma 12. Let $\Phi = \bigvee_{j=1}^{m} \Psi_j$ be a 3-cnf formula with m disjunctive 3-clauses. Then, Φ has a 1-in-3 model iff

$$\min_{M \in \mathcal{MO}(k)} \delta_1(S_\Phi,\ M \cdot T_\Phi) \quad = \quad m \cdot (36 + 4\sqrt{2}).$$

Furthermore, if Φ has no 1-in-3 model then

$$\min_{M \in \mathcal{MO}(k)} \delta_1(S_\Phi,\ M \cdot T_\Phi) \quad \geqslant \quad m \cdot (36 + 4\sqrt{2}) + (6 - 4\sqrt{2}).$$

Proof. Assume that α is a 1-in-3 model of Φ, i. e., $\alpha \models \Psi_j$ for each $1 \leqslant j \leqslant m$. Then, let $M_B \coloneqq \eta^{-1}(\alpha)$. By Corollary 1 and the definition of $B_{\Psi_j}(M_B)$ from Lemma 11, $\delta_{\Psi_j}(M_B) = 36 + 4\sqrt{2}$ holds for each $1 \leqslant j \leqslant k$, and thus

$$\begin{aligned} \delta_1(S_\Phi, M_B \cdot T_\Phi) &= \sum_{1 \leqslant j \leqslant m} \delta_1(S_{\Psi_j}, M_B \cdot T_{\Psi_j}) \\ &= m \cdot (36 + 4\sqrt{2}). \end{aligned}$$

Note that, by Corollary 1, $\delta_1(S_\Phi, M_B \cdot T_\Phi)$ cannot be any smaller, i. e.,

$$\min_{M \in \mathcal{MO}(k)} \delta_1(S_\Phi,\ M \cdot T_\Phi) = m \cdot (36 + 4\sqrt{2}).$$

For the other direction, assume that there is no 1-in-3 model α for Φ. Now, choose an arbitrary assignment α' and an arbitrary $M \in \mathcal{MO}(k)$ with $\hat{\eta}(M') = \alpha'$. Since α' is no 1-in-3 model of Φ, it is no 1-in-3 model for at least one disjunctive 3-clause Ψ_j of Φ. By Corollary 1 and the definition of $B_{\Psi_j}(M')$ from Lemma 11, $\delta_{\Psi_j}(M') \geqslant 42$ holds. For the other disjunctive 3-clauses in Φ, Corollary 1 claims a lower bound of $36 + 4\sqrt{2}$; thus,

$$\begin{aligned} \delta_1(S_\Phi,\ M' \cdot T_\Phi) &= \sum_{1 \leqslant j \leqslant m} \delta_1(S_{\Psi_j},\ M' \cdot T_{\Psi_j}) \\ &\geqslant (m-1) \cdot (36 + 4\sqrt{2}) + 42. \end{aligned} \tag{4.14}$$

Recall that $\hat{\eta}$ partitions $\mathcal{MO}(k)$. The proof of this lemma is complete since Inequation 4.14 holds for arbitrary assignments α' and orthogonal matrices $M' \in \hat{\eta}^{-1}(\alpha)$. □

Note that Lemma 12 establishes the goal formulated directly before equation (4.8): When choosing

$$c^{\mathcal{MO}}(\Phi) \quad \coloneqq \quad m \cdot (36 + 4\sqrt{2})$$

whenever Φ is a 3-cnf formula consisting of m disjunctive 3-clauses, then Lemma 12 claims that Φ has a 1-in-3 model if, and only if, $\delta_1^{\mathcal{MO}}(S_\Phi, T_\Phi) = c^{\mathcal{MO}}(\Phi)$. Furthermore, $\delta_1^{\mathcal{MO}}(S_\Phi, T_\Phi) \geqslant c^{\mathcal{MO}}(\Phi) + (6 - 4\sqrt{2})$ whenever Φ has no 1-in-3 model, which states a clear gap of $6 - 4\sqrt{2} \approx 0.34$ between the two cases.

Relating $\delta_1^C(\bar{S}_\Phi, \bar{T}_\Phi)$ with 1-in-3 models of Φ: The former results only considered transformations using orthogonal matrices. However, the congruence distance δ_1^C allows distance minimization also by translating with an arbitrary vector. The following lemma considers these transformations of time series too.

Lemma 13. Let S and T be two time series of the same length over $\mathbb{R}^k$ and let $\bar{S} := S \times -S$ and $\bar{T} := T \times -T$. The following is true for every $M \in \mathcal{MO}(k)$ and every $v \in \mathbb{R}^k$:

$$\delta_1(\bar{S},\ M \cdot \bar{T}) \quad \leqslant \quad \delta_1(\bar{S},\ M \cdot \bar{T} + v)$$

Proof. Let S, T, v be as in the lemma's assumption. Fixate an arbitrary $M \in \mathcal{MO}(k)$. Furthermore, let $T' := M \cdot T$, denote $S = (s_0, \cdots, s_{n-1})$, and denote $T' = (t_0, \cdots, t_{n-1})$. Then,

$$\begin{aligned} \delta_1\left(\bar{S}, M \cdot \bar{T}\right) &= \left(\sum_{i=0}^{n-1} \mathsf{d}_2(s_i, t_i)\right) + \left(\sum_{i=0}^{n-1} \mathsf{d}_2(-s_i, -t_i)\right) \\ &= \sum_{i=0}^{n-1} 2 \cdot \|s_i - t_i\|_2 \,, \end{aligned}$$

and

$$\delta_1\left(\bar{S}, M \cdot \bar{T} + v\right) = \sum_{i=0}^{n-1} \left(\|s_i - t_i - v\|_2 + \|s_i - t_i + v\|_2 \right).$$

Letting $u_i \coloneqq s_i - t_i$ yields

$$\begin{aligned} \delta_1(\bar{S}, M \cdot \bar{T}) &= \sum_{i=0}^{n-1} 2\,\|u_i\|_2 \,, \qquad \text{and} \\ \delta_1(\bar{S}, M \cdot \bar{T} + v) &= \sum_{i=0}^{n-1} \Big(\|u_i + v\|_2 + \|u_i - v\|_2 \Big). \end{aligned}$$

For proving the lemma, it therefore suffices to show that

$$2\,\|u_i\|_2 \quad \leqslant \quad \|u_i + v\|_2 + \|u_i - v\|_2 \tag{4.15}$$

is true for every $i \in [0, n)$. In the following, Inequality (4.15) is proven in fact to be true for *every* vector $u_i \in \mathbb{R}^k$. To that, the following claim will be useful.

Claim 1. $|a+b|+|a-b| \geqslant 2|a|$ *is true for all* $a, b \in \mathbb{R}$.

Proof of Claim 1: First, restrict attention to the case where $a \geqslant 0$. In this case, the following is true:

$$|a+b|+|a-b| = \begin{cases} a+b+a-b &= 2a, & \text{if } a \geqslant |b| \\ a+|b|-a+|b| &= 2|b|, & \text{if } a < |b| \end{cases}$$

In both cases, $|a+b|+|a-b| \geqslant 2|a|$, and thus the proof is done for $a \geqslant 0$.

Now, consider the case where $a < 0$. Then, $a' := -a > 0$ yields $|a+b|+|a-b| = |a'-b|+|a'+b| \geqslant 2|a'| = 2|a|$ with the case study from above, which completes the proof of Claim 1. □

Now, let u be an arbitrary vector in $\mathbb{R}^k$. Clearly, an $M' \in \mathcal{MO}(k)$ exists such that $M'u = ae_1$ for some $a \in \mathbb{R}$. For this M', let $b := M'v$; thus, b_1 is the first component of the vector $M'v$. Then, the following is true:

$$2\,\|u\|_2 \;=\; 2\left\|M'u\right\|_2 \;=\; 2|a|$$

and

$$\begin{aligned} \|u+v\|_2 + \|u-v\|_2 &= \left\|M'(u+v)\right\|_2 + \left\|M'(u-v)\right\|_2 \\ &= \left\|M'u+M'v\right\|_2 + \left\|M'u-M'v\right\|_2 \\ &= \left\|ae_1+M'v\right\|_2 + \left\|ae_1-M'v\right\|_2 \\ &\geqslant |a+b_1| \;+\; |a-b_1|. \end{aligned}$$

By Claim 1, the inequation continues with $\|u+v\|_2 + \|u-v\|_2 \;\geqslant\; 2|a|$. In summary,

$$2\,\|u\|_2 \;=\; 2|a| \;\leqslant\; \|u+v\|_2 \;+\; \|u-v\|_2$$

is true for all $u, v \in \mathbb{R}^k$, which completes the proof of Lemma 13. □

Finally, the following theorem relates 3-cnf formulas Φ with the congruence distance of the corresponding time series $\bar{S}_\Phi$ and $\bar{T}_\Phi$.

Theorem 4. Let $\Phi = \bigvee_{j=1}^{m} \Psi_j$ be a 3-cnf formula with m disjunctive clauses. Then, Φ has a 1-in-3 model iff

$$\delta_1^C(\bar{S}_\Phi,\ \bar{T}_\Phi) \quad = \quad m \cdot (72 + 8\sqrt{2}).$$

Furthermore, if Φ has no 1-in-3 model then

$$\delta_1^C(\bar{S}_\Phi,\ \bar{T}_\Phi) \quad \geqslant \quad m \cdot (72 + 8\sqrt{2}) + (12 - 8\sqrt{2}).$$

Proof. Lemma 13 yields

$$\delta_1^C(\bar{S}_\Phi, \bar{T}_\Phi) \;=\; \min_{M \in \mathcal{MO}(k)} \delta_1(\bar{S}_\Phi,\ M \cdot \bar{T}_\Phi).$$

Furthermore, by the choice of $\bar{S}_\Phi$ and $\bar{T}_\Phi$,

$$\delta_1(\bar{S}_\Phi,\ M \cdot \bar{T}_\Phi) \;=\; 2 \cdot \delta_1(S_\Phi,\ M \cdot T_\Phi).$$

holds for every $M \in \mathcal{MO}(k)$. Hence, by Lemma 12, Φ has a 1-in-3 model iff

$$\begin{aligned}\delta_1^C(\bar{S}_\Phi,\ \bar{T}_\Phi) &= 2 \cdot \min_{M \in \mathcal{MO}(k)} \delta_1(S_\Phi,\ M \cdot T_\Phi) \\ &= 2 \cdot m \cdot (36 + 4\sqrt{2}).\end{aligned}$$

If, on the other hand, Φ has no 1-in-3 model then, by Lemma 12,

$$\begin{aligned}\delta_1^C(\bar{S}_\Phi,\ \bar{T}_\Phi) &= 2 \cdot \min_{M \in \mathcal{MO}(k)} \delta_1(S_\Phi,\ M \cdot T_\Phi) \\ &\geqslant 2 \cdot m \cdot (36 + 4\sqrt{2}) + 2 \cdot (6 - 4\sqrt{2}).\end{aligned}$$

□

An algorithm solving 1-in-3-Sat: The proof of Theorem 3 reduces the 1-IN-3-SAT problem to δ_1^C-COMPUTATION. This reduction uses Theorem 4, which only considers 3-cnf formulas consisting of no trivial disjunctive 3-clauses. Therefore, the reduction of Theorem3 needs a transformation of a 3-cnf formula

$$\Phi' = \bigwedge_{j \in I} \Psi'_j$$

with trivial disjunctive 3-clauses to a 3-cnf formula with no trivial disjunctive 3-clauses in polynomial time.

To that, consider the trivial 3-clause

$$\Psi'_{j^*} \;=\; V_{i_1} \vee \neg V_{i_1} \vee L_{i_2}$$

for some $i_1, i_2 \in [1, k]$. Clearly, removing Ψ'_{j^*} from Φ' is feasible in polynomial time; and Φ' is equivalent to $\Phi \coloneqq \bigwedge_{j \in I \setminus \{j^*\}} \Psi_j$.

For a trivial disjunctive 3-clause of the form

$$\Psi'_{j^*} \;=\; L_{i_1} \vee L_{i_1} \vee L_{i_2}$$

where L_{i_1} is a literal over a variable V_i, consider a new variable V_0 that is not in Φ'. Then, Ψ'_{j^*} is equivalent to the following 3-cnf formula:

$$\Phi_1 \;\coloneqq\; \Big(L_{i_1} \vee L_{i_2} \vee V_0\Big) \wedge \Big(L_{i_1} \vee L_{i_2} \vee \neg V_0\Big)$$

Clearly, Φ' and

$$\Phi \;\coloneqq\; \Phi_1 \wedge \bigwedge_{j \in I \setminus \{j^*\}} \Psi'_j$$

are equivalent.

Analogously, for a trivial 3-clause of the form

$$\Psi'_{j^*} \;=\; L_{i_1} \vee L_{i_1} \vee L_{i_1}$$

consider two new variables V_0 and V'_0 that are not in Φ'. Then, Ψ_{j^*} is equivalent to

$$\begin{aligned}\Phi_2 \;\coloneqq\; &\left(L_{i_1} \vee V_0 \vee V'_0\right) \wedge \left(L_{i_1} \vee V_0 \vee \neg V'_0\right) \\ &\wedge \left(L_{i_1} \vee \neg V_0 \vee V'_0\right) \wedge \left(L_{i_1} \vee \neg V_0 \vee \neg V'_0\right)\end{aligned}$$

and Φ' and

$$\Phi \;\coloneqq\; \Phi_2 \wedge \bigwedge_{j \in I \setminus \{j^*\}} \Psi'_j$$

are equivalent.

If multiple trivial disjunctive 3-clauses have to be replaced in Φ', the new variables V_0 and V'_0 can be reused, i. e., the transformation adds at most two new variables to the 3-cnf formula while iteratively applying the transformation rules above. Furthermore, the length of the 3-cnf formula extends at most by a factor of four. Clearly, this reduction is feasible in polynomial time regarding the length of the 3-cnf formula Φ'.

Proof of Theorem 3. Assume that $\mathcal{A}$ is an algorithm that, upon the input of the dimensionality k and two time series S and T of equal length, computes $\delta_1^C(S,T)$. The problem 1-IN-3-SAT can be solved by using this algorithm as follows.

First, apply the transformation above to the input 3-cnf formula Φ' to gain a 3-cnf formula Φ that only consists of non-trivial disjunctive 3-clauses. Then, construct the time series $\bar{S}_\Phi$ and $\bar{T}_\Phi$. Clearly, this is feasible in polynomial time regarding the size of Φ'. Letting k be the number of variables occurring in Φ, run the algorithm $\mathcal{A}$ with input k, $\bar{S}_\Phi$, and $\bar{T}_\Phi$. After a number of steps polynomial in the size of Φ, $\mathcal{A}$ will output the number $\delta_1^C(S,T)$ (up to a precision of $12 - 8\sqrt{2} \approx 0.69$). Now, check if this number is equal to (a suitable representation of) the number $m \cdot (72 + 8\sqrt{2})$ where m is the number of disjunctive 3-clauses of Φ. If so, output "yes"; otherwise, output "no".

Theorem 4 yields that the algorithm's output is "yes" if, and only if, Φ (and therefore Φ') has a 1-in-3 model. Thus, the construction of a

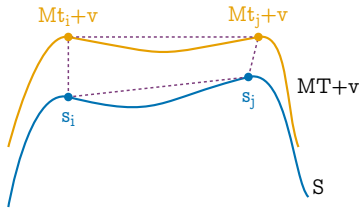


Figure 4.3: A sketch of a time series S and a time series T that is transformed by $M \in \mathcal{MO}(k)$ and $v \in \mathbb{R}^k$. The dashed lines represent distances.

polynomial-time algorithm solving the NP-complete problem 1-IN-3-SAT is done. Such an algorithm, however, cannot exist in case that P $\neq$ NP. □

Note that according to the proof above, already the restriction of the problem δ_1^C-COMPUTATION to input time series over

$$\{0, 1, -1, 6, -6\}^k$$

cannot be accomplished in polynomial time, unless P = NP.

The numbers computed in the proof of Theorem 3 are to be computed up to a precision of $12 - 8\sqrt{2} \approx 0.69$. This precision is constant for all input sizes. Thus, the precision of the floating-point numbers needs to grow logarithmically with the input size.

4.3 Approximating the Congruence Distance

Consider two time series S and T of equal length and an isometric transformation consisting of an orthogonal matrix $M \in \mathcal{MO}(k)$ and a vector $v \in \mathbb{R}^k$ as sketched in Figure 4.3. The approximations proposed in this section all derive from the following estimation: The triangle inequality yields

$$\begin{aligned} \mathtt{d}(s_i, s_j) &\leqslant \mathtt{d}(s_i, Mt_i + v) + \mathtt{d}(Mt_i + v, Mt_j + v) + \mathtt{d}(Mt_j + v, s_j) \\ &= \mathtt{d}(s_i, Mt_i + v) + \mathtt{d}(t_i, t_j) + \mathtt{d}(Mt_j + v, s_j) \end{aligned}$$

and

$$\begin{aligned} \mathtt{d}(t_i, t_j) &= \mathtt{d}(Mt_i + v, Mt_j + v) \\ &\leqslant \mathtt{d}(s_i, Mt_i + v) + \mathtt{d}(s_i, s_j) + \mathtt{d}(Mt_j + v, s_j) \end{aligned}$$

and thus for each $0 \leqslant i, j < \#S$:

$$|\mathtt{d}(s_i, s_j) - \mathtt{d}(t_i, t_j)| \leqslant \mathtt{d}(s_i, Mt_i + v) + \mathtt{d}(s_j, Mt_j + v). \tag{4.16}$$

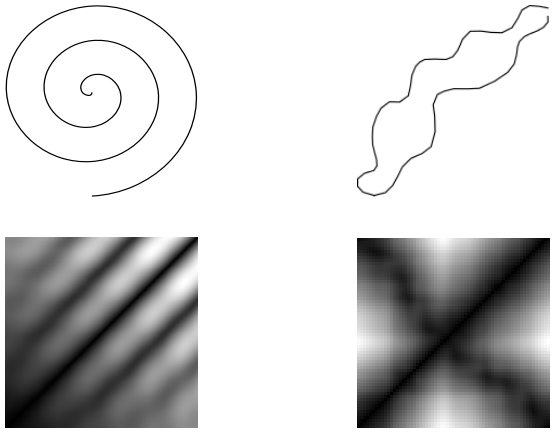


Figure 4.4: A sketch of two time series S (top left) and T (top right) and a visualization of their respective self-similarity matrices ΔS (bottom left) and ΔT (bottom right). In the matrices, the intensity corresponds to the distance values, i. e., dark pixels represent small distance values and light pixels represent larger distance values.

The right hand side of Inequation 4.16 consists of elements from the congruence distance $\delta_1^C(S,T) = \sum_{i=0}^{n-1} \mathtt{d}(s_i, Mt_i + v)$. Since the left hand side of Inequation 4.16 is independent of the transformation used on the right hand side, this inequation helps estimating the congruence distance without actually solving an optimization problem (cf. Section 4.1.2 for an interpretation of $\delta_p^C(S,T)$ as optimization problem). Using this insight, this section proposes two approximating algorithms in Section 4.3.1 and 4.3.2.

To formalize the approximations of the congruence distance, the well-known[7] *self-similarity matrix* of time series is used.

Definition 10 (Self-Similarity Matrix)**.** The self-similarity matrix of an arbitrary time series $T = (t_0, \ldots, t_{n-1})$ is the matrix

$$\Delta T \;\coloneqq\; \big(\, \mathtt{d}\,(t_i, t_j)\, \big)_{0 \leqslant i,j < n}.$$

Note that $\Delta T_{i,j} = \Delta T_{j,i}$ and $\Delta T_{i,i} = 0$ holds for each $0 \leqslant i, j < n$. Figure 4.4 sketches two time series and a visualization of their respective self-similarity matrices. The diagonals of the self-similarity matrices are black because $\mathtt{d}(t_i, t_i) = 0$ holds for each t_i in a time series T. Furthermore, the matrices are symmetric because the distance function $\mathtt{d}$ is symmetric.

Considering the self-similarity matrices ΔS and ΔT, the left-hand side of Inequation (4.16) matches the entries of $|\Delta S - \Delta T|$. The important property that makes the self-similarity matrix useful for approximating the

[7] Usually, the self-similarity matrix is used to analyze a time series for patterns (e. g., using Recurrence Plots [39]).

congruence distance is its invariance under isometric transformations, i. e., the self-similarity matrix ΔT completely describes the sequence T up to congruence:

Theorem 5. Consider the metric space $(\mathbb{R}^k, \mathtt{d}_2)$, and let S, T be two time series of length n over $\mathbb{R}^k$. Then, S and T are congruent iff they have the same self-similarity matrix, i. e.,

$$S \cong_C T \quad \iff \quad \Delta S = \Delta T. \tag{4.17}$$

Intuitively, Theorem 5 holds because the Euclidean Distance is invariant under isometric transformations. Still, the proof is provided for the sake of completeness.

Lemma 14. Let $\mathcal{B} = \{b_1, \cdots, b_n\}$ be a basis of a vector space $V \subseteq \mathbb{R}^k$ and $u, v \in V$. Then,

$$\forall i \in [1, n] : \langle u, b_i \rangle \;=\; \langle v, b_i \rangle \quad \iff \quad u = v.$$

Proof. Clearly, $\langle u, w \rangle = \langle v, w \rangle$ holds for all $w \in \mathbb{R}^k$ if $u = v$.

The interesting direction is shown by induction over n: For $\mathcal{B} = \{b_1\}$, the statement is trivial. For $n > 1$, consider $\mathcal{B} = \{b_1, \cdots, b_n\}$ and let

$$\begin{aligned} u' \;&\coloneqq\; u - \langle u, b_n \rangle \cdot \frac{1}{\|b_n\|_2^2} \cdot b_n \qquad \text{and} \\ v' \;&\coloneqq\; v - \langle v, b_n \rangle \cdot \frac{1}{\|b_n\|_2^2} \cdot b_n. \end{aligned}$$

Then,

$$\begin{aligned} \langle u', b_n \rangle \;&=\; \left\langle u - \langle u, b_n \rangle \cdot \frac{1}{\|b_n\|_2^2} \cdot b_n,\; b_n \right\rangle \\ &=\; \langle u, b_n \rangle \;-\; \langle u, b_n \rangle \cdot \frac{1}{\|b_n\|_2^2} \cdot \langle b_n, b_n \rangle \\ &=\; \langle u, b_n \rangle \;-\; \langle u, b_n \rangle \\ &=\; 0 \end{aligned}$$

holds, i. e., $u' \in \mathit{span}(b_1, \cdots, b_{n-1})$. Analogously, $v' \in \mathit{span}(b_1, \cdots, b_{n-1})$ holds. By induction, $u' = v'$ holds, and thus

$$\begin{aligned} u - \langle u, b_n \rangle \cdot \frac{1}{\|b_n\|_2^2} \cdot b_n \;&=\; u' \\ &=\; v' \\ &=\; v \;-\; \langle v, b_n \rangle \cdot \frac{1}{\|b_n\|_2^2} \cdot b_n. \end{aligned}$$

With $\langle u, b_n \rangle = \langle v, b_n \rangle$, the induction step $u = v$ is complete. □

Proof of Theorem 5. Let $S = (s_0, \cdots, s_{n-1})$ and $T = (t_0, \cdots, t_{n-1})$. For the direction "$\Longrightarrow$", assume that $S \cong_C T$, i. e., $T = M \cdot S + v_0$ for some orthogonal matrix $M \in \mathcal{MO}(k)$ and some vector $v_0 \in \mathbb{R}^k$. Then, $\Delta S = \Delta T$ since the following equation holds for all $i, j \in [0, n)$:

$$\begin{aligned}\mathtt{d}(t_i, t_j) &= \mathtt{d}_2(Ms_i + v_0,\ Ms_j + v_0)\\ &= \|Ms_i + v_0 - Ms_j - v_0\|_2\\ &= \|M(s_i - s_j)\|_2\\ &= \|s_i - s_j\|_2\\ &= \mathtt{d}_2(s_i, s_j)\end{aligned}$$

In the opposite direction, assume $\Delta S = \Delta T$. In order to find the transforming matrix $M \in \mathcal{MO}(k)$, let $S' = (0, s'_1, \cdots, s'_{n-1}) = S - s_0$ and $T' = (0, t'_1, \cdots, t'_{n-1}) = T - t_0$. Then, $\Delta S' = \Delta S = \Delta T = \Delta T'$, and thus $\mathtt{d}_2(s'_i, s'_j) = \mathtt{d}_2(t'_i, t'_j)$ holds for $i, j \in [1, n)$. Therefore, the equality

$$\langle s'_i - s'_j,\ s'_i - s'_j\rangle \ =\ \mathtt{d}_2(s'_i, s'_j)^2 \ =\ \mathtt{d}_2(t'_i, t'_j)^2 \ =\ \langle t'_i - t'_j,\ t'_i - t'_j\rangle,$$

and thus

$$\begin{aligned}\langle s'_i, s'_i\rangle + \langle s'_j, s'_j\rangle - 2\langle s'_i, s'_j\rangle &= \langle s'_i - s'_j, s'_i - s'_j\rangle\\ &= \langle t'_i - t'_j, t'_i - t'_j\rangle\\ &= \langle t'_i, t'_i\rangle + \langle t'_j, t'_j\rangle - 2\langle t'_i, t'_j\rangle\end{aligned}$$

holds for all $i, j \in [1, n)$. Using further $\langle s'_i, s'_i\rangle = \mathtt{d}_2(0, s'_i)^2 = \mathtt{d}_2(0, t'_i)^2 = \langle t'_i, t'_i\rangle$ for all $i \in [1, n)$, the equality of the scalar products

$$\langle s'_i, s'_j\rangle \ =\ \langle t'_i, t'_j\rangle \tag{4.18}$$

follows. Now, Equation (4.18) helps defining the function which yields the desired matrix $M \in \mathcal{MO}(k)$. To that, let $\mathcal{B} \coloneqq \{s'_{i_1}, \cdots, s'_{i_m}\}$ be a basis of the vector space

$$V \ \coloneqq\ \mathit{span}(s'_1, \cdots, s'_{n-1})$$

and let $W \coloneqq \mathit{span}(t'_{i_1}, \cdots, t'_{i_m})$ (using the same indices as in $\mathcal{B}$). It is easy to construct a function $F' : V \longrightarrow W$ with $F'(s'_{i_j}) \coloneqq t'_{i_j}$ for each $j \in [1, m]$. Because of Equation (4.18), the equality

$$\langle F'(s'_{i_a}), F'(s'_{i_b})\rangle \ \ =\ \ \langle t'_{i_a}, t'_{i_b}\rangle \ \ =\ \ \langle s'_{i_a}, s'_{i_b}\rangle$$

holds for all $a, b \in [1, m]$; thus, F' is an orthogonal function.Therefore,

$$\begin{aligned}\langle t'_j, t'_{i_a}\rangle &= \langle s'_j, s'_{i_a}\rangle\\ &= \langle F'(s'_j), F'(s'_{i_a})\rangle\\ &= \langle F'(s'_j), t'_{i_a}\rangle\end{aligned}$$

holds for all $j \in [1, n)$ and $a \in [1, m]$. Now, Lemma 14 yields $F'(s'_j) = t'_j$ for $j \in [1, n)$.

With basic linear algebra methods, F' can be extended to an orthogonal function $F : \mathbb{R}^k \longrightarrow \mathbb{R}^k$. Consider the matrix $M \in \mathcal{MO}(k)$ such that

$$F(v) = w \quad \Longleftrightarrow \quad Mv = w$$

is true for all $v \in \mathbb{R}^k$. In particular, $M \cdot s'_i = t'_i$ is true for all $i \in [1, n)$, and thus $M \cdot S' = T'$. Hence,

$$T - t_0 \;=\; T' \;=\; M \cdot S' \;=\; M \cdot (S - s_0) \;=\; M \cdot S - s_0,$$

and therefore $T = M \cdot S + (t_0 - s_0)$, i. e., $S \cong_C T$. □

4.3.1 Metric Approximation

Equation (4.16) and (4.17) motivate the approach for approximating the congruence: A distance function on the self-similarity matrices of two time series S and T estimates their congruence distance. This approach yields a (pseudo) metric distance function.

Definition 11 (Delta Distance)**.** Let S, T be two time series of length n. The *delta distance* $\delta^\Delta(S, T)$ is

$$\delta^\Delta(S, T) \coloneqq \frac{1}{2} \max_{0<r<n} \sum_{i=0}^{n-1} \left| \mathtt{d}_2 \left(s_i, s_{(i+r) \mod n}\right) - \mathtt{d}_2 \left(t_i, t_{(i+r) \mod n}\right) \right|.$$

Proposition 2. The delta distance δ^Δ satisfies the triangle inequality.

Proof. Consider three time series S, T, and U and fixate an r^* that maximizes $\delta^\Delta(R, T)$ in Definition 11. Then,

$$\begin{aligned}
\delta^\Delta(S, U) \;&=\; \frac{1}{2} \sum_{i=0}^{n-1} \left| \mathtt{d}_2 \left(s_i, s_{(i+r^*) \mod n}\right) - \mathtt{d}_2 \left(u_i, u_{(i+r^*) \mod n}\right) \right| \\
&\leqslant\; \frac{1}{2} \sum_{i=0}^{n-1} \Big(\left| \mathtt{d}_2 \left(s_i, s_{(i+r^*) \mod n}\right) + \mathtt{d}_2 \left(t_i, t_{(i+r^*) \mod n}\right) \right| + \\
&\qquad\qquad \left| \mathtt{d}_2 \left(t_i, t_{(i+r^*) \mod n}\right) - \mathtt{d}_2 \left(u_i, u_{(i+r^*) \mod n}\right) \right| \Big) \\
&\leqslant\; \delta^\Delta(S, T) + \delta^\Delta(T, U)
\end{aligned}$$

proves the triangle inequality. □

Since $\mathtt{d}_2$ is symmetric, δ^Δ inherits its symmetry. Hence, δ^Δ is a pseudo metric on the set of time series of length n. Furthermore, δ^Δ is a metric on the set of equivalence classes regarding the congruence relation $\cong_C$.

The complexity for computing the delta distance δ^Δ in Definition 11 grows quadratically with the time series' length.

The next aim is to show that the the delta distance δ^Δ provides a lower bound for the congruence distance δ_1^C:

Theorem 6. For all time series S and T, the following inequation holds:

$$\delta^\Delta(S,T) \quad \leqslant \quad \delta_1^C(S,T).$$

Proof. Fixate a range r^* which maximizes $\delta^\Delta(S,T)$ in Definition 11. Using the triangle inequality as in Equation (4.16) yields

$$\begin{aligned}
\delta^\Delta(S,T) &= \frac{1}{2}\sum_{i=0}^{n-1} \left|\mathtt{d}_2\left(s_i, s_{(i+r^*) \mod n}\right) - \mathtt{d}_2\left(t_i, t_{(i+r^*) \mod n}\right)\right| \\
&= \frac{1}{2}\sum_{i=0}^{n-1} \Big|\mathtt{d}\left(s_i, s_{(i+r^*) \mod n}\right) \\
&\qquad\qquad - \mathtt{d}_2\left(M\cdot t_i + v, M\cdot t_{(i+r^*) \mod n} + v\right)\Big| \\
&\leqslant \frac{1}{2}\sum_{i=0}^{n-1} \Big(\mathtt{d}\left(s_i, M\cdot t_i + v\right) \\
&\qquad\qquad + \mathtt{d}_2\left(s_{(i+r^*) \mod n}, M\cdot t_{(i+r^*) \mod n} + v\right)\Big) \\
&= \sum_{i=0}^{n-1} \mathtt{d}_2\left(s_i, M\cdot t_i + v\right) \\
&= \delta_1^C(S,T)
\end{aligned}$$

for arbitrary $M \in \mathcal{MO}(k)$ and $v \in \mathbb{R}^k$. □

On the one hand, Theorem 6 shows that the delta distances δ^Δ provides a lower bound for the congruence distance δ_1^C. On the other hand, the ratio of the congruence distance and the delta distance can grow arbitrarily, as shown with the following example.

Example 6. Consider $\mathbb{M} = \mathbb{R}^2$. The following shows that time series S, T exist for each $C > 0$ such that $\frac{\delta^C(S,T)}{\delta^\Delta(S,T)} \geqslant C$ (cf. Figure 4.5). Let $\varepsilon > 0$, $a \coloneqq \sqrt{1-\varepsilon^2}$, and consider $S_\varepsilon = (s_0, s_1, s_2), T_\varepsilon = (t_0, t_1, t_2)$ with

$$S_\varepsilon \coloneqq \left(\begin{pmatrix}-a\\0\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}a\\0\end{pmatrix}\right) \quad , \quad T_\varepsilon \coloneqq \left(\begin{pmatrix}-a\\0\end{pmatrix}, \begin{pmatrix}0\\\varepsilon\end{pmatrix}, \begin{pmatrix}a\\0\end{pmatrix}\right).$$

Then, $\delta^\Delta(S_\varepsilon, T_\varepsilon) = 2(1-a) = 2(1-\sqrt{1-\varepsilon^2})$. If $\delta_1^C(S_\varepsilon, T_\varepsilon) \geqslant \frac{\varepsilon}{2}$ then

$$\frac{\delta_1^C(S_\varepsilon, T_\varepsilon)}{\delta^\Delta(S_\varepsilon, T_\varepsilon)} \geqslant \frac{1}{4}\cdot\frac{\varepsilon}{1-\sqrt{1-\varepsilon^2}} \xrightarrow{\varepsilon\to 0} +\infty.$$

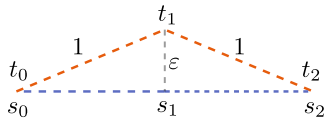


Figure 4.5: Two time series S_ε and T_ε in $\mathbb{R}^2$ with $\frac{d_1^C(S_\varepsilon,T_\varepsilon)}{d_1^\Delta(S_\varepsilon,T_\varepsilon)} \xrightarrow{\varepsilon\to 0} +\infty$.

It remains to prove that $\delta_1^C(S_\varepsilon, T_\varepsilon) \geqslant \frac{\varepsilon}{2}$. To that, *assume* that $M \in \mathcal{MO}(k)$ and $v_0 \in \mathbb{M}$ exist such that

$$\delta_1(S_\varepsilon, M \cdot T_\varepsilon + v_0) < \frac{\varepsilon}{2}.$$

Then, considering the linear function

$$\begin{aligned} \iota : \mathbb{M} &\longrightarrow \mathbb{M} \\ v &\longmapsto M \cdot v + v_0, \end{aligned}$$

the following inequality must hold for each $i \in [0, 2]$:

$$\mathtt{d}_2(s_i, \iota(t_i)) < \frac{\varepsilon}{2}$$

Since ι is an isometric function and $\frac{1}{2}s_0 + \frac{1}{2}s_2 - s_1 = 0$,

$$\begin{aligned} \mathtt{d}_2\left(s_1, \frac{\iota(t_0)}{2} + \frac{\iota(t_2)}{2}\right) &= \left\| \frac{\iota(t_0)}{2} + \frac{\iota(t_2)}{2} - s_1 \right\|_2 \\ &= \left\| \frac{\iota(t_0)}{2} - \frac{s_0}{2} + \frac{\iota(t_2)}{2} - \frac{s_2}{2} + \frac{s_0}{2} + \frac{s_2}{2} - s_1 \right\|_2 \\ &\leqslant \frac{1}{2}\mathtt{d}_2(\iota(t_0), s_0) + \frac{1}{2}\mathtt{d}_2(\iota(t_2), s_2) \\ &< \frac{\varepsilon}{2}, \end{aligned}$$

holds, i. e., a contradiction rises with

$$\begin{aligned} \mathtt{d}(s_1, \iota(t_1)) &\geqslant \underbrace{\mathtt{d}_2\left(\iota(t_1), \iota\left(\frac{1}{2}(t_0 + t_2)\right)\right)}_{=\mathtt{d}_2\left(t_1, \frac{1}{2}(t_0+t_2)\right)=\varepsilon} - \mathtt{d}_2\left(s_1, \iota\left(\frac{1}{2}(t_0 + t_2)\right)\right) \\ &> \varepsilon - \frac{\varepsilon}{2}. \end{aligned}$$

Hence, $\delta_1^C(S_\varepsilon, T_\varepsilon) = \min_{M\in\mathcal{MO}(k), v_0\in\mathbb{M}} \delta_1(S_\varepsilon, M \cdot T_\varepsilon + v_0) \geqslant \frac{\varepsilon}{2}$.

This subsection provided the delta distance that is a metric lower bound for the congruence distance with quadratic runtime in the length and linear runtime in the time series' dimensionality.

4.3.2 Greedy Approximation

While the delta distance sums up values along each (wrapped) diagonal in $|\Delta S - \Delta T|$ and chooses the largest of these sums, another combination of elements within $|\Delta S - \Delta T|$ as summands might provide a better approximation of the congruence distance. Since it is a computationally expensive task to try all combinations, the following approach tries to find a good combination using a greedy algorithm for selecting the entries of $|\Delta S - \Delta T|$.

The greedy algorithm first sorts the elements $d_{i,j} = |\mathtt{d}_2(s_i, s_j) - \mathtt{d}_2(t_i, t_j)|$ in descending order and stores them in a sequence $Q = (d_{i_1,j_1}, d_{i_2,j_2}, \cdots)$. While iterating over the sequence Q, it adds d_{i_r,j_r} to a global sum and masks the indices i_r and j_r as already seen. In further iterations, the algorithm skips elements in the queue that access already seen indices; thus, each index is used at most once. Basically, this is why the greedy delta distance (denoted as $\delta^G(S,T)$) is a lower bound for the congruence distance. Theorem 7 proves this claim. Algorithm 11 provides the pseudo-code for the computation of δ^G.

Algorithm 11 Greedy Delta Distance

```
1  Algorithm: greedydelta
2  Input: time series S,T of length n
3  Output: distance d
4
5  let Q = () // empty sequence
6  for i = 0,...,n-2
7      for j = i+1,...,n-1
8          append d_{i,j} := |d_2(s_i,s_j) - d_2(t_i,t_j)| to Q
9  sort Q // (descending)
10 let S = ∅
11 let d = 0
12 for each d_{i_a,j_a} in Q
13     if i_a ∈ S or j_a ∈ S continue
14     let d = d + d_{i_a,j_a}
15     let S = S ∪ {i_a, j_a}
16 return d
```

The complexity is dominated by sorting n^2 elements, which in turn takes $n^2 \cdot \log(n^2)$ steps.

Theorem 7. For all time series S and T, the following inequation holds:

$$\delta^G(S,T) \quad \leqslant \quad \delta_1^C(S,T).$$

Proof. Let $Q^* = (d_{i_1,j_1}, \cdots, d_{i_r,j_r})$ be the list of elements from the queue in Algorithm 11 which have not been skipped. Since each index is appears at most once in this list, the following inequality holds for arbitrary orthogonal matrices M and vectors $v \in \mathbb{R}^k$:

$$\begin{aligned}
\delta^G(S,T) &= \sum_{a=1}^{r} d_{i_a,j_a} \\
&= \sum_{a=1}^{r} \left| \mathtt{d}_2\left(s_{i_a}, s_{j_a}\right) - \mathtt{d}_2\left(M \cdot t_{i_a} + v, M \cdot t_{j_a} + v\right) \right| \\
&\leqslant \sum_{a=1}^{r} \Big(\mathtt{d}_2\left(s_{i_a}, M \cdot t_{i_a} + v\right) + \mathtt{d}_2\left(s_{j_a}, M \cdot t_{j_a} + v\right) \Big) \\
&\leqslant \sum_{i=0}^{n-1} \mathtt{d}_2\left(s_i, M \cdot t_i + v\right)
\end{aligned}$$

Hence, $\delta^G(S,T) \leqslant \delta_1^C(S,T)$. □

4.3.3 Runtime Improvement

The delta distance's and greedy delta distance's complexity are linear regarding the dimensionality but quadratic regarding the length. This section motivates an optimization for both algorithms.

Time series usually do not contain random points but come from continuous processes in the real world, i. e., the distance between two successive elements is relatively small compared to the distance of two elements that are far away in time. Hence, the distances $\mathtt{d}\left(t_i, t_j\right)$ and $\mathtt{d}\left(t_i, t_{j+1}\right)$ are probably relatively close to each other if $i \ll j$ (i. e., if j is much larger than i). This insight leads to the idea to only consider elements $\mathtt{d}\left(t_i, t_j\right)$ where $|i-j|$ is a power of two, i. e., less elements are considered for larger temporal distances.

The Fast Delta Distance: Adapting the above described idea to the delta distance δ^Δ yields the following definition.

Definition 12 (Fast Delta Distance)**.** Let S, T be two time series of length n. The *fast delta distance* $\tilde{\delta}^\Delta(S,T)$ is

$$\tilde{\delta}^\Delta(S,T) \coloneqq \frac{1}{2} \max_{0<\delta<n, \delta \in 2^{\mathbb{N}}} \sum_{i=0}^{n-1} \left| \mathtt{d}_2\left(s_i, s_{(i+\delta) \mod n}\right) - \mathtt{d}_2\left(t_i, t_{(i+\delta) \mod n}\right) \right|.$$

Since $\tilde{\delta}^\Delta$ only omits values in comparison to δ^Δ (cf. Definition 11), the fast version $\tilde{\delta}^\Delta$ is a lower bound for δ^Δ:

Corollary 2. For all time series S and T of equal length, the following inequation holds:

$$\tilde{\delta}^{\Delta}(S,T) \quad \leqslant \quad \delta^{\Delta}(S,T)$$

In particular, the fast delta distance also is a lower bound for the congruence distance. For time series of length n, the complexity of the fast delta distance $\tilde{\delta}^{\Delta}$ improves to $n \log n$. On the down side, equivalence classes regarding the fast delta distance might include time series which are not congruent, as the following example shows.

Example 7. Let

$$S = \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 3 \\ 0 \end{pmatrix}, \begin{pmatrix} 3 \\ 4 \end{pmatrix}, \begin{pmatrix} 2 \\ 2 \end{pmatrix} \right) \quad , \quad T = \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 3 \\ 0 \end{pmatrix}, \begin{pmatrix} 3 \\ 4 \end{pmatrix}, \begin{pmatrix} 4 \\ 2 \end{pmatrix} \right).$$

Then,

$$\Delta S = \begin{pmatrix} 0 & 3 & 5 & \sqrt{8} \\ 3 & 0 & 4 & \sqrt{5} \\ 5 & 4 & 0 & \sqrt{5} \\ \sqrt{8} & \sqrt{5} & \sqrt{5} & 0 \end{pmatrix} \quad , \quad \Delta T = \begin{pmatrix} 0 & 3 & 5 & \sqrt{20} \\ 3 & 0 & 4 & \sqrt{5} \\ 5 & 4 & 0 & \sqrt{5} \\ \sqrt{20} & \sqrt{5} & \sqrt{5} & 0 \end{pmatrix}$$

and $\delta^{\Delta}(S,T) = \sqrt{20} - \sqrt{8}$, but $\tilde{\delta}^{\Delta}(S,T) = 0$ because the fast version does not consider $\mathsf{d}(s_0, s_3)$ and $\mathsf{d}(t_0, t_3)$.

The fast greedy delta distance: Incorporating the idea for improving the runtime into the greedy delta distance changes Line 7 of Algorithm 11: Only values for the variable j are considered that add a power of 2 to the variable i. Algorithm 12 provides the pseudo-code for the fast greedy delta distance $\tilde{\delta}^{G}$.

Example 8. Let

$$S = \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 36 \\ 2 \end{pmatrix}, \begin{pmatrix} 36 \\ -2 \end{pmatrix}, \begin{pmatrix} 72 \\ 0 \end{pmatrix} \right) \quad , \quad T = \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 40 \\ 2 \end{pmatrix}, \begin{pmatrix} 40 \\ -2 \end{pmatrix}, \begin{pmatrix} 80 \\ 0 \end{pmatrix} \right).$$

With $\sqrt{36^2 + 2^2} \approx 36.06$ and $\sqrt{40^2 + 2^2} \approx 40.05$, the similarity matrices of S and T are

$$\Delta S \approx \begin{pmatrix} 0 & 36.06 & 36.06 & 72 \\ 36.06 & 0 & 4 & 36.06 \\ 36.06 & 4 & 0 & 36.06 \\ 72 & 36.06 & 36.06 & 0 \end{pmatrix} \quad \text{and}$$

$$\Delta T \approx \begin{pmatrix} 0 & 40.05 & 40.05 & 80 \\ 40.05 & 0 & 4 & 40.05 \\ 40.05 & 4 & 0 & 40.05 \\ 80 & 40.05 & 40.05 & 0 \end{pmatrix},$$

Algorithm 12 Fast Greedy Delta Distance

```
1  Algorithm: fastgreedydelta
2  Input: time series S,T of length n
3  Output: distance d
4
5  let Q = () // empty sequence
6  for i = 0,...,n-2
7      for j = i+1, i+2, i+4, ..., max{j* ∈ 2^ℕ | j* ⩽ n-1}
8          append d_{i,j} := |d_2(s_i, s_j) - d_2(t_i, t_j)| to Q
9  sort Q // (descending)
10 let S = ∅
11 let d = 0
12 for each d_{i_a,j_a} in Q
13     if i_a ∈ S or j_a ∈ S continue
14     let d = d + d_{i_a,j_a}
15     let S = S ∪ {i_a, j_a}
16 return d
```

respectively. Therefore, $\delta^G(S,T) = 8$ and $\tilde{\delta}^G(S,T) \approx ¸8.02$. It is easy to find examples where $\tilde{\delta}^G(S,T) \leqslant \delta^G(S,T)$. Sill, this example shows that analogous statements to Corollary 2 for $\tilde{\delta}^G$ do not hold, i. e., time series S and T exist with $\tilde{\delta}^G(S,T) > \delta^G(S,T)$.

The fast greedy delta distance $\tilde{\delta}^G$ is again dominated by the sorting of elements. This time, $n \log n$ elements have to be sorted, thus its complexity is $n \log(n) \log(n \log n)$ which asymptotically behaves as $n \log(n)^2$. Hence, the fast versions both have quasi-linear runtime regarding length and linear runtime regarding dimensionality.

An inequality such as in Theorem 2 does not exist for the fast greedy delta distance (cf. Example 8). Also, there is no correlation between the (fast) delta distance and the (fast) greedy distance. The evaluation in Section 4.4 shows that the greedy delta distance provides a much better approximation in most cases. Also, Section 4.4 evaluates the tightness of the proposed approximations to the congruence distance.

4.4 Evaluation

The approximations' evaluation is of higher interest than the evaluation of the congruence distance itself since the exact computation of the congruence distance is a computationally hard problem and thus not feasible in practical applications. Unfortunately, there is no direct algorithm for the computation of the congruence distance; thus, a numerical optimizer is

necessary for the computation of the congruence distance interpreted as a nonlinear optimization problem (cf. Section 4.1.2). For two time series S and T, the distance value computed by such an optimizer is denoted with $\delta^O(S,T)$. Since an optimizer might not find the global optimum, all values for the congruence distance (computed by an optimizer) in this section are in fact upper bounds to the correct but unknown value of the congruence distance, i. e., $\delta_1^C(S,T) \leqslant \delta^O(S,T)$. This given circumstance complicates the evaluation of the approximations to the congruence distance.

To deal with that problem, a first step is to estimate the error of the optimizer regarding different hyperparameters (e. g., dimensionality and length of time series) by evaluating it on pairs of time series for which the congruence distance is known (cf. Section 4.4.1). For those hyperparameters that yield a small error, the later estimation of the approximation's tightness is assumed to be accurate. On the other hand, for hyperparameters that yield large errors, the estimation of the approximation's tightness is assumed to be loose, i. e., the approximation is probably tighter than the experiments claim. Unfortunately, providing explicit statements is not feasible.

For a detailed explanation, consider a lower bound $\ell(S,T)$ to the congruence distance (e. g., ℓ might be one of δ^G, $\tilde{\delta}^G$, δ^Δ, or $\tilde{\delta}^\Delta$) and suppose $\delta^O(S,T) = \delta_1^C(S,T) + \varepsilon$, i. e., $\varepsilon \geqslant 0$ is the error of the optimizer. Then, the following correlation between the estimated tightness and the real tightness holds:

$$1 \geqslant \frac{\ell(S,T)}{\delta_1^C(S,T)} = \frac{\ell(S,T)}{\delta^O(S,T) - \varepsilon} \geqslant \frac{\ell(S,T)}{\delta^O(S,T)}$$

Hence, for small errors ε, the estimated tightness is accurate and for large errors ε the tightness is underestimated. Section 4.4.2 and 4.4.3 evaluate the tightness and the approximations speedup to the (optimizer based) congruence distance, respectively.

4.4.1 Congruence Distance: An Optimization Problem

Consider fixed time series S and T in $\mathbb{R}^k$ with length n. The congruence distance is a nonlinear optimization problem with equality based constraints. The function to minimize is

$$f\,(M,v) = \sum_{i=0}^{n-1} \mathtt{d}\,(s_i, M \cdot t_i + v)$$

while the k^2 equality based constraints correspond to the constraints for orthogonal matrices:

$$M \cdot M^T = I.$$

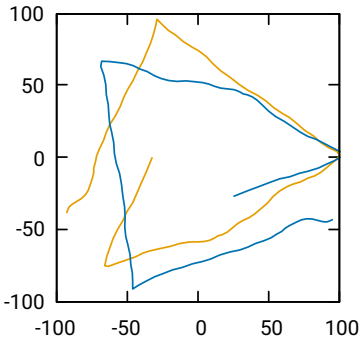


Figure 4.6: Sketch of two two-dimensional congruent time series.

To construct a problem for that the result is known, a random orthogonal matrix M^* and a small random vector v^* transforms a time series T. The optimizer is then urged to solve the optimization problem $\delta_1^C(T, M^* \cdot T + v^*)$. Clearly, the optimizer needs to find a solution with value 0 since both time series are congruent, indeed. Here, the optimizer is considered not to work properly whenever it claims large distance values.

In this thesis' scope, different optimizer strategies have been tried. An augmented lagrangian algorithm [22] with the BOBYQA algorithm [52] as local optimizer has been chosen for further experiments because it promised the best performance with these experiments. The implementation is based on the algorithms provided by the NLopt library [55].

The first experiment evaluates the congruence distance on four datasets with dimensionality 1 to 4, respectively. Each of the datasets consists of 200 `RAM` generated time series of length 100 (and bounding sphere radius 100) and 1 transformed version for each. Figure 4.6 shows an example of a generated time series and its transformed version.

In total, the experiment conducts 800 optimization problems as described above. The further evaluation discards the samples for that no reasonable solution was found by the optimizer. Figure 4.7 shows the distance values proposed by the optimizer (and therefore the error it makes) per dimensionality. For experiments with dimensionality 5 or higher, the optimizer failed to find any reasonable value near 0 within 24 hours of computation time. Figure 4.7 also shows that the computation time rapidly increases with increasing dimensionality. Because of the rising error and runtime with increasing dimensionality, an evaluation of the congruence distance on higher dimensionality is not feasible. Hence, all further evaluation only considers up to 4-dimensional time series.

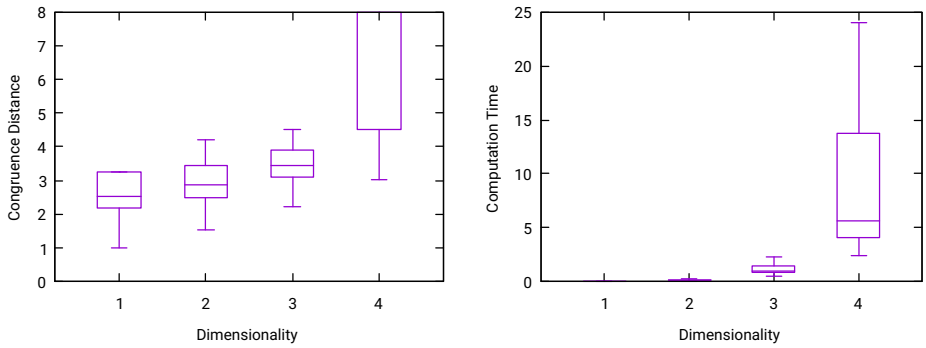


Figure 4.7: Boxplot: distance values (left) and runtime (right) of the optimizer on congruent time series.

4.4.2 Tightness of Approximations

To evaluate the tightness of the (fast) delta distance and (fast) greedy delta distance to the congruence distance, the experiments in this section use the `RAM` dataset generator and the real-world Character Trajectories dataset (CT) that contains over 2800 two-dimensional time series [67]. Other real-world datasets with higher dimensionality are not suitable because, on those, the optimizer fails to compute reasonable results of the congruence distance.

Since making the (greedy) delta distance time-warping aware is future work, another way is necessary to deal with time-warping here. To that, the CT dataset is preprocessed such that each time series, seen as a trajectory, moves with constant speed, i. e., for each *dewarped* time series, the following holds for all suitable indices i and j:

$$\mathtt{d}_2\left(t_i, t_{i+1}\right) \approx \mathtt{d}_2\left(t_j, t_{j+1}\right).$$

This property is achieved by reinterpolating the time series regarding the arc length.

Figure 4.8 shows the tightness of the approximations on the synthesized datasets. As expected, the greedy delta distance provides the tightest approximation to the congruence distance (provided by the optimizer used in this work).

As observed in Section 4.4.1, the optimizer's error increases with increasing dimensionality. Hence, the tightness of the optimizer to the actual congruence distance is decreasing. Since a similar behavior can be observed here (the tightness of the approximation is decreasing with increasing dimensionality), the reason might be the optimizer's inaccuracy. Either way, the tightness is above 50% in most cases. Especially when using the greedy delta distance, the tightness is above 75% in most cases.

On the CT dataset, the delta distance and the greedy delta distance achieved a tightness of 63% and 83%, respectively.

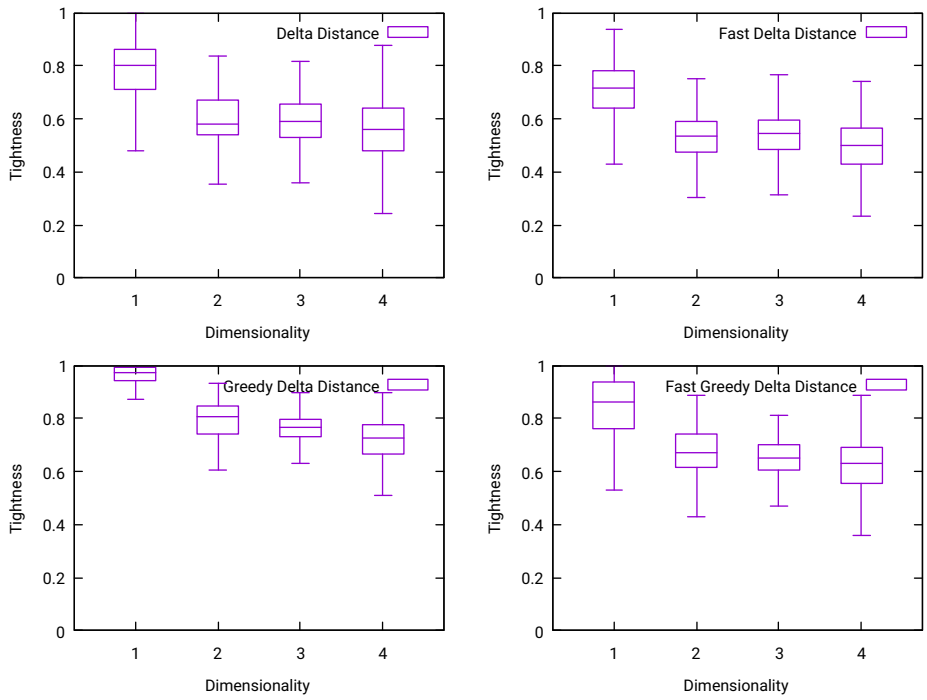


Figure 4.8: Average tightness of the delta distance (top left), the fast delta distance (top right), the greedy delta distance (bottom left), and the fast greedy delta distance (bottom right) to the congruence distance, respectively.

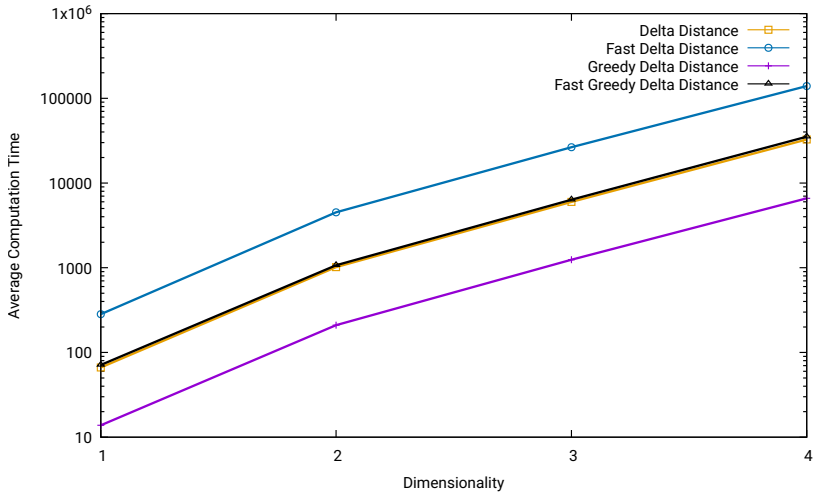


Figure 4.9: Average speedup of the approximations to the optimizer.

4.4.3 Speedup of Approximations

Figure 4.9 shows the speedup of the approximations to the optimizer. As expected, the speedup increases exponentially with increasing dimensionality. While the fast delta distance is the fastest algorithm, it also provides the worst approximation (compare with Figure 4.8). On the other hand, the greedy delta distance provides the best approximation while being the slowest algorithm. Still, the greedy delta distance is multiple orders of magnitudes faster than our optimizer.

The approximating algorithms achieved the following speedups on the character trajectory dataset: 1642 with the delta distance; 8040 with the fast delta distance; 321 with the greedy delta distance; 2287 with the fast greedy delta distance. The results are similar to those on the `RAM` generated datasets.

4.4.4 Conclusion

This chapter analyzed the problem of measuring the congruence between two time series. It showed that the computation of the congruence distance is not (and will never be) feasible. On the other hand, it provides four approximating distance functions that are at least two orders of magnitude faster than the congruence distance itself, one of which is suitable for metric index structures (delta distance), and one of which loses this benefit but seems to achieve a tighter approximation (greedy delta distance). The delta distance and greedy delta distance algorithms have linear complexity regarding the dimensionality but quadratic worst-case runtime complexity regarding the time series length. The other two approximations address this problem at the cost of approximation quality; they have quasi-linear runtime regarding the length.

Chapter 5

Subsequence Search using Metric Index Structures

A common approach to accelerating similarity search algorithms is the usage of index structures. In the case of metric distance functions, metric index structures such as the M-Tree are applicable. On the other hand, many applications ask for approximate subsequences or subsets, e. g., searching for a similar partial sequence of a gene, a similar scene in a movie, or a similar object in a picture represented by a set of multi-dimensional features. Metric index structures cannot be utilized for these tasks because subsequence search is not symmetric, but metric distance functions are.

This chapter proposes an extension of the M-Tree, the SuperM-Tree, that allows approximate subsequence and subset queries as nearest neighbor queries. The SuperM-Tree indexes metric subset spaces, a new, generalized concept of metric spaces. Various metric distance functions are extendable to metric subset distance functions, e. g., the Euclidean distance (on subsequences), the Hausdorff distance (on subsets), the Edit distance, and the Dog Keeper distance (on subsequences); these examples subsume the applications mentioned above.

5.1 Introduction

The ubiquitous B^+-Tree [36] has its home in relational database management systems; it provides fast range queries on linearly ordered data. The R^*-Tree [16] is the standard index structure for indexing spatial data; it offers the capability of fast multi-dimensional range queries. Numerous index structures exist that offer more expressive queries on more complex data types; this includes index structures for sets and set containment joins [53, 63, 87], for strings and similarity search regarding the Edit distance [84], and for nearest neighbor queries in any metric space [19,23,35,76].

Metric index structures are more generic than the ones mentioned above.

They support range queries on a wide range of different complex data types, such as the Euclidean distance on sequences of the same length, the Edit distance (ED) on strings, the Dog Keeper distance (DK) on sequences or multi-dimensional trajectories of arbitrary lengths [9, 10], and the (two-sided) Hausdorff distance on sets (e. g., on sets of integers or sets of feature vectors) [38]. However, the symmetry of a metric strongly constrains the queries' expressiveness and prevents containment queries that ask for subsequences, subsets, and the like.

An index structure that supports approximate subset queries needs to discard the symmetry, and it needs a relation regarding the size of the objects. Hence, this chapter introduces the *metric subset space* $(\mathcal{M}, d, \sqsubseteq)$ consisting of a set of objects $\mathcal{M}$, a distance function d, and a total preorder $\sqsubseteq$ ("smaller than or equal size") ordering the objects by their size. The concept of the metric subset space discards the symmetry and the reflexivity of d and reduces the necessity of the triangle inequality such that the triangle inequality $d(x, z) \leqslant d(x, y) + d(y, z)$ only needs to hold for objects $x, y, z \in \mathcal{M}$ with $x \sqsubseteq y \sqsubseteq z$.

Inspired by the M-Tree, this chapter proposes the SuperM-Tree that indexes metric subset spaces. The SuperM-Tree supports k-nearest neighbor and ε-nearest neighbor queries: Given a query object q, the result of a *sub query* consists of objects p, such that $p \sqsubseteq q$ (p is not larger than q) and p is similar to a part of q (regarding the semantic of the metric subset distance function). As a demonstration of the generality of metric subset spaces, this chapter provides several examples, including approximate set containment queries and subsequence queries with various distance functions, such as the Euclidean distance (on subsequences), the Hausdorff distance (on subsets), and the Dog Keeper distance (on subsequences).

In summary, this chapter proposes a general index structure for database systems that supports natural types of queries for a wide range of multimedia data types.

The rest of this chapter is structured as follows: Section 5.2 describes the concept of metric subset spaces and demonstrates it on examples. Section 5.3 introduces the data structure of the SuperM-Tree and the algorithms for insertion and searching of objects. Section 5.4 evaluates the efficiency of the SuperM-Tree on three different subset distance functions. The evaluation shows that the speedup against a linear scan search algorithm increases with increasing size of the dataset.

Note that this chapter is considered to be a proof of concept. Hence, not all algorithms required in real-world applications are discussed here, e. g., this thesis avoids the algorithms for deleting objects from the SuperM-Tree, since they are not necessary for evaluating the performance of the SuperM-Tree.

5.2 Metric Subset Spaces

Consider a set of objects $\mathcal{M}$, for example, sequences of arbitrary lengths or sets of multi-dimensional vectors. Natural queries put objects in a certain "containment" relationship when they ask for subsets or subsequences. Here, a total preorder (denoted with '$\sqsubseteq$') describes this containment relationship. Intuitively, this relationship requires each pair of objects (x, y) that x is either not larger or not smaller than y (or both). Note that the preorder does not consider similarity at all.

For an exact subsequence T' of T with $d(T', T) = 0$, the reflexivity would claim that $T' = T$, which prohibits the existence of exact subsequences. Hence, discarding the reflexivity is essential. Since the order of the parameters of the distance function declares the subsequence relation (e. g., $d(S, T)$ is the distance of S to T with S being a subsequence), discarding the symmetry is essential.[1]

Given an exact subsequence S of T and U (at the same time), S might match T and U at different positions, e. g., S is the prefix of T and the suffix of U. The triangle inequality claims that $d(T, U) \leqslant d(S, T) + d(S, U) = 0$, which is a contradiction to the possibility that $T \neq U$. Therefore, the concept of the metric subset space restricts the demand for the triangle inequality to hold on transitive chains of the preorder, i. e., $d(x, z) \leqslant d(x, y) + d(y, z)$ if $x \sqsubseteq y \sqsubseteq z$. Of course, the triangle inequality can hold for other triplets x, y, z, but this is no must.

The following definitions formalize the idea motivated above.

Definition 13 (Total Preorder)**.** A total preorder on $\mathcal{M}$ is a relation $\sqsubseteq$ on $\mathcal{M} \times \mathcal{M}$ that satisfies the following axioms:

$$\begin{aligned} \forall x \in \mathcal{M} :&\ \ x \sqsubseteq x \\ \forall x, y \in \mathcal{M} :&\ \ x \sqsubseteq y \ \vee\ y \sqsubseteq x \\ \forall x, y, z \in \mathcal{M} :&\ \ x \sqsubseteq y \ \wedge\ y \sqsubseteq z \ \longrightarrow\ x \sqsubseteq z \end{aligned}$$

Note that a total preorder defines an equivalence relation $\equiv$ by

$$x \equiv y \iff x \sqsubseteq y \ \wedge\ y \sqsubseteq x.$$

Examples for a total preorder include comparing the length of time series or the cardinality of finite sets.

Definition 14 (Metric Subset Space)**.** A metric subset space is a 3-tuple $(\mathcal{M}, \sqsubseteq, d)$ consisting of a set $\mathcal{M}$, a total preorder $\sqsubseteq$ on $\mathcal{M}$, and a function $d : \mathcal{M} \times \mathcal{M} \to \mathbb{R}^{\geqslant 0}$ that satisfies the following axiom:

$$\mathcal{S}) \qquad \forall x, y, z \in \mathcal{M} :\ x \sqsubseteq y \sqsubseteq z \ \longrightarrow\ d(x, z) \leqslant d(x, y) + d(y, z)$$

[1] Remark that two the time series S and T are in subsequence relation in both directions (with more or less similarity), e. g., when considering time-warping distance functions.

Compared to metric spaces, Axiom $\mathcal{S}$ relaxes the triangle inequality; and it discards the reflexivity and symmetry. Considering the trivial total preorder that relates all objects (i. e., $\forall x, y : x \sqsubseteq y$), each metric space also is a metric subset space; thus, metric subset spaces generalize metric spaces.

The following Corollary 3 claims that switching the parameters in the total preorder and the distance function yields another metric subset space. In other words, the following corollary converts subset ordering to superset ordering and vice versa.

Corollary 3. For a metric subset space $(\mathcal{M}, \sqsubseteq, d)$, let $y \sqsupseteq x :\Longleftrightarrow x \sqsubseteq y$ and $\tilde{d}(y, x) \coloneqq d(x, y)$ for all $x, y \in \mathcal{M}$. Then, $(\mathcal{M}, \sqsupseteq, \tilde{d})$ is a metric subset space.

The following three examples (that are typical applications) show the generality of metric subset spaces.

5.2.1 Euclidean Distance on Subsequences

Let $\mathcal{M}$ be the set of real valued time series and let $S = (s_0, \cdots, s_{m-1}), T = (t_0, \cdots, t_{n-1}) \in \mathcal{M}$ be such two time series with $m = \#S$ and $n = \#T$. The canonical total preorder for subsequence search is the "shorter or equal than" relation, i. e., $S \sqsubseteq T :\Longleftrightarrow \#S \leqslant \#T$. For sequences with $S \sqsubseteq T$, a common approach for subsequence search is the windowing approach with the Euclidean distance [42]:[2]

$$\delta_2(S, T) \;\coloneqq \min_{0 \leqslant j \leqslant n-m} \delta_2(S, T_{j,m}).$$

The following proposition shows that this model for subsequence distances yields a metric subset space.

Proposition 3. $(\mathcal{M}, \sqsubseteq, \delta_2)$ is a metric subset space.

The proposition even holds for subsequences of elements from another metric space (e. g., n-dimensional vectors).

Proof. Consider three time series

$$\begin{aligned} S &= (s_0, \cdots, s_{m-1}), \\ T &= (t_0, \cdots, t_{n-1}), \text{ and} \\ U &= (u_0, \cdots, u_{k-1}) \end{aligned}$$

with $S \sqsubseteq T \sqsubseteq U$ and fix $s, t \in \mathbb{N}$ such that

$$\begin{aligned} \delta_2(S, T) &= \delta_2\left(S,\ T_{s,m}\right) \quad \text{and} \\ \delta_2(T, U) &= \delta_2\left(T,\ U_{t,n}\right). \end{aligned}$$

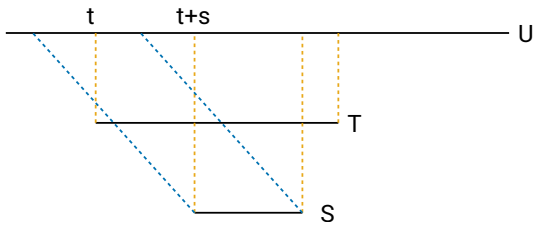


Figure 5.1: Sketch for the variables choice in the proof of Proposition 3. The striped lines indicate the best matches of the subsequences (yellow: S in T, T in U; blue: S in U).

Figure 5.1 sketches this situation. The following three inequalities help to prove the triangle inequality.

1: Since δ_2 is a metric on common length sequences, the following triangle inequality holds:

$$\delta_2(S, U_{t+s,m}) \leqslant \delta_2(S, T_{s,m}) + \delta_2(T_{s,m}, U_{t+s,m}) \tag{5.1}$$

2: The monotonicity of δ_2 regarding the length yields the second inequality:

$$\begin{aligned} \delta_2(T_{s,m}, U_{t+s,m})^2 &= \sum_{s \leqslant i < s+m} \mathtt{d}(T_i, U_{t+i})^2 \\ &\leqslant \sum_{i=0}^{n-1} \mathtt{d}(T_i, U_{t+i})^2 \\ &= \delta_2(T, U_{t,n})^2 \end{aligned} \tag{5.2}$$

3: The last inequality uses the properties of the windowing approach, which is scanning for the best match:

$$\delta_2(S, U) = \min_{0 \leqslant j \leqslant k-m} \delta_2(S,\ U_{j,m}) \leqslant \delta_2(S,\ U_{t+s,m}) \tag{5.3}$$

Combining inequalities (5.1),(5.2), and (5.3) completes the proof:

$$\begin{aligned} \delta_2(S, U) &\leqslant \delta_2(S,\ U_{t+s,m}) \\ &\leqslant \delta_2(S,\ T_{s,m}) + \delta_2(T_{s,m},\ U_{t+s,m}) \\ &\leqslant \delta_2(S,\ T_{s,m}) + \delta_2(T,\ U_{t,n}) \\ &= \delta_2(S, T) + \delta_2(T, U) \end{aligned}$$

□

[2]Recall that $T_{i,l}$ denotes the subsequence of T starting at index i with length l.

5.2.2 Dog Keeper Distance on Subsequences

Again, let $\mathcal{M}$ be a set of real valued time series; let $S, T, U \in \mathcal{M}$ with

$$\begin{aligned} S &= (s_0, \cdots, s_{m-1}), \\ T &= (t_0, \cdots, t_{n-1}), \quad \text{and} \\ U &= (u_0, \cdots, u_{k-1}); \end{aligned}$$

and let $\sqsubseteq$ be the length comparison.

Since the Dog Keeper distance DK is able to stretch the "time" axis, the method presented here generalizes the windowing approach to windows of arbitrary length:

$$\texttt{S-DK}(S, T) \coloneqq \min_{\substack{0 \leqslant j < n \\ 1 \leqslant \ell \leqslant n-j}} \texttt{DK}\left(S,\ T_{j,\ell}\right)$$

The difference to the windowing approach in Section 5.2.1 is that the length of the subsequence in T is not bound to the length of S. The computation of S-DK has quadratic runtime complexity since it is a simple modification of the DK distance similar to modifying DTW to achieve its subsequence version S-DTW [73].

The subsequence distance function S-DK yields a metric subset space.

Proposition 4. $(\mathcal{M}, \sqsubseteq, \texttt{S-DK})$ is a metric subset space.

Proof. Similar to Proposition 3, it suffices to prove the triangle inequality for arbitrary $S, T, U \in \mathcal{M}$ with $S \sqsubseteq T \sqsubseteq U$. To that, fix $s_a, s_\ell, t_a, t_\ell \in \mathbb{N}$ such that

$$\begin{aligned} \texttt{S-DK}(T, U) &= \texttt{DK}\left(T,\ U_{t_a,t_\ell}\right) \quad \text{and} \\ \texttt{S-DK}(S, T) &= \texttt{DK}\left(S,\ T_{s_a,s_\ell}\right). \end{aligned}$$

Figure 5.2 sketches this situation.

Recall that the computation of $\texttt{DK}\left(T, U_{t_a,t_\ell}\right)$ maps elements of T to elements of U; thus, it maps each subsequence of T to a certain subsequence of U. In particular, u_a and u_ℓ exist such that the subsequence T_{s_a,s_ℓ} is mapped to U_{u_a,u_ℓ}. Since this is a part of the mapping of T to U, the inequality

$$\texttt{DK}\left(T_{s_a,s_\ell}, U_{u_a,u_\ell}\right) \leqslant \texttt{DK}\left(T, U_{t_a,t_\ell}\right) \tag{5.4}$$

holds. For these subsequences, the triangle inequality of the metric distance function DK [9] is applicable. Thus,

$$\texttt{DK}\left(S, U_{u_a,u_\ell}\right) \leqslant \texttt{DK}\left(S, T_{s_a,s_\ell}\right) + \texttt{DK}\left(T_{s_a,s_\ell}, U_{u_a,u_\ell}\right). \tag{5.5}$$

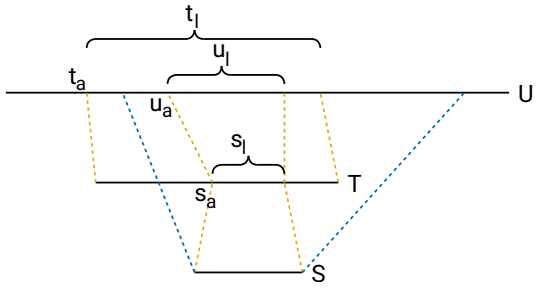


Figure 5.2: Sketch for the variables choice in the proof of Proposition 4. The striped lines indicate the best matches of the subsequences (yellow: S in T, T in U; blue: S in U).

Together with Inequation (5.4) and by the choice of s_a, s_ℓ, t_a, and t_ℓ, this yields the triangle inequality for S-DK:

$$\begin{aligned}
\texttt{S-DK}(S,U) &\leqslant \texttt{DK}\left(S, U_{u_a,u_\ell}\right) \\
&\leqslant \texttt{DK}\left(S, T_{s_a,s_\ell}\right) + \texttt{DK}\left(T_{s_a,s_\ell}, U_{u_a,u_\ell}\right) \\
&\leqslant \texttt{DK}\left(S, T_{s_a,s_\ell}\right) + \texttt{DK}\left(T, U_{t_a,t_\ell}\right) \\
&= \texttt{S-DK}\left(S,T\right) + \texttt{S-DK}\left(T,U\right)
\end{aligned}$$

□

5.2.3 Hausdorff Distance on Subsets

To provide an example for metric subset spaces from another data domain, let the members of $\mathcal{M}$ be sets of real values. Here, the total preorder compares the cardinalities of sets $A, B \in \mathcal{M}$, i. e., $A \sqsubseteq B \;:\Longleftrightarrow\; \#A \leqslant \#B$. The Hausdorff distance is a common metric distance function on sets:

$$\textsc{Hausdorff}(A,B) \;\coloneqq\; \max\left\{ \max_{a\in A}\min_{b\in B}|a-b|,\; \max_{b\in B}\min_{a\in A}|a-b| \right\}$$

In more general, the Hausdorff distance is a metric for sets of arbitrary elements from any other metric space (e. g., Euclidean vector spaces). This example sticks to sets of real values to maintain readability.

In the case of subset comparison, the symmetry of the Hausdorff distance is not necessary. The following definition derived from the Hausdorff distance yields a subset distance function (S-HD) with similar semantics:

$$\texttt{S-HD}(A,B) \;\coloneqq\; \max_{a\in A}\min_{b\in B}|a-b|$$

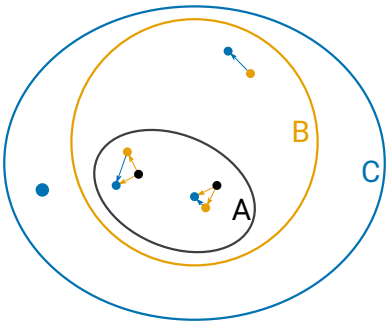


Figure 5.3: Sketch for illustrating the proof of Proposition 5. The arrows indicate the mappings from A to B, from B to C, and from A to C.

While S-HD tries to match A to a subset of B, the original HAUSDORFF distance does this in both directions, i. e.,

$$\text{HAUSDORFF}(A, B) \;=\; \max\Big\{\texttt{S-HD}(A, B),\;\; \texttt{S-HD}(B, A)\Big\}.$$

Proposition 5. $(\mathcal{M}, \sqsubseteq, \texttt{S-HD})$ is a metric subset space.

Proof. As in the proofs of Proposition 3 and 4, it suffices to prove the triangle inequality for arbitrary $A, B, C \in \mathcal{M}$ with $A \sqsubseteq B \sqsubseteq C$. Therefore, fixate mappings $f : A \longrightarrow B$ and $g : B \longrightarrow C$ such that

$$\begin{aligned} \forall a \in A:\; f(a) \;&=\; \arg\min_{b \in B} |a - b| \quad \text{and} \\ \forall b \in B:\; f(b) \;&=\; \arg\min_{c \in C} |b - c|, \end{aligned}$$

i. e., f and g map all elements to their corresponding nearest neighbor. Hence,

$$\begin{aligned} \texttt{S-HD}(A, B) \;&=\; \max_{a \in A} |a - f(a)| \quad \text{and} \\ \texttt{S-HD}(B, C) \;&=\; \max_{b \in B} |b - g(b)|. \end{aligned}$$

Now, for arbitrary $a \in A$, the inequation

$$\begin{aligned} \min_{c \in C} |a - c| \;&\leqslant |a - g(f(a))| \\ &\leqslant |a - f(a)| \;+\; |f(a) - g(f(a))| \\ &\leqslant \texttt{S-HD}(A, B) \;+\; \texttt{S-HD}\,(f(A), C) \\ &\leqslant \texttt{S-HD}(A, B) \;+\; \texttt{S-HD}(B, C) \end{aligned}$$

holds, and thus $\texttt{S-HD}(A, C) \;\leqslant\; \texttt{S-HD}(A, B) \;+\; \texttt{S-HD}(B, C)$. □

5.3 The SuperM-Tree

The SuperM-Tree is derived from the M-Tree [35] and differs due to indexing metric subset spaces $(\mathcal{M}, \sqsubseteq, d)$ instead of metric spaces. The basic properties are the same (cf. Section 2.2.3): It is a balanced tree with nodes of a specific capacity (either of fixed size or variable size as in [17]). Inner nodes contain *routing objects* that cover a nearby part of the metric subset space. The covered objects include all objects in the corresponding subtree. Leaf nodes contain the actual entries.

Additionally, the SuperM-Tree needs the objects along each path from the root to a leaf to be ordered according to the metric subset space's total preorder. That way, the data structure assures the preliminary of the triangle inequality in metric subset spaces, i. e., the data structure assures that the triangle inequality holds to prune certain subtrees. Consider, for example, the tree in Figure 2.3. There, the relations $B \sqsubseteq D$, $B \sqsubseteq X$, and $D \sqsubseteq X$ need to hold. On the other hand, $C \sqsubseteq X$ or $X \sqsubseteq D$ is not necessary.

Also, the insert and delete algorithms (including the split and merge strategies) need adjustment to keep the ordering condition. This section provides a proof of concept for the SuperM-Tree. To that, the deletion algorithms are not necessary and therefore not included here.

For the rest of this section, let $(\mathcal{M}, \sqsubseteq, d)$ be an arbitrary metric subset space.

5.3.1 Structure of SuperM-Tree Nodes

Since the structure of the SuperM-Tree is derived from the M-Tree [35], this thesis uses the same notation. Leaf nodes store the indexed (key) objects, whereas internal nodes store a set of *routing objects* that help in pruning subtrees and navigating through the tree.

Each routing object O_r consists of

- $T(O_r)$, the *covered subtree*, i. e., a reference to the root node of its subtree;
- $r(O_r)$, the *covering radius*; and
- $d\big(O_r, P(O_r)\big)$, the distance of O_r to its parent node $P(O_r)$ (or nil if O_r is in the root node of the tree).

In addition to the M-Tree, the SuperM-Tree expects that $O_r \sqsubseteq O_j$ for each object O_j in its covered tree. All indexed objects in the covered tree of O_r are within the distance $r(O_r)$ from O_r.

Leaf nodes store a set of indexed objects O_j and their distance to the parent $d\big(P(O_j), O_j\big)$, respectively. In real-world scenarios, leaf nodes store additional information per object, e. g., a data pointer or tuple identifier.

5.3.2 Similarity Queries

The SuperM-Tree supports range queries and nearest neighbor queries. This section only discusses range queries since nearest neighbor queries are simple range queries that adapt the search radius while traversing the tree, analogously to the M-Tree.

For an object $Q \in \mathcal{M}$, a *range query* selects all database objects that are in the search area and that are not larger than the query object, i. e.,

$$NN\big(Q, r(Q)\big) := \big\{O_j \in \mathcal{M} \quad | \quad d(O_j, Q) \leqslant r(Q) \ \wedge \ O_j \sqsubseteq Q\big\}$$

Algorithm 13 provides the pseudo-code for the range query. The algorithm uses Lemma 15 and 16 to prune subtrees.[3]

Algorithm 13 Range query on SuperM-Tree

```
1  Algorithm: supermtree.range
2  Input: node N (default: root node), query object Q,
       search radius r(Q)
3  R := ∅
4  if N is a leaf node
5    for each O_j ∈ N
6      if O_j ⊑ Q and d(O_j, Q) ⩽ r(Q)
7        R := R ∪ {O_j}
8    return R
9  // else
10 for each O_r in N
11   if not O_r ⊑ Q
12     skip // prune subtree by ''size''
13   if r(Q) < d(P(O_r), Q) − d(O_r, P(O_r)) − r(O_r)
14     skip // subtree pruned using Lemma 16
15   if r(Q) < d(O_r, Q) − r(O_r)
16     skip // subtree pruned using Lemma 15
17   R := R ∪ supermtree.range(T(O_r), Q, r(Q))
18 return R
```

Lemma 15. If $d(O_r, Q) > r(Q) + r(O_r)$, then $d(O_j, Q) > r(Q)$ holds for each object O_j with $O_j \sqsubseteq Q$ in the covered subtree of O_r.

Remark that, since only objects O_j with $O_j \sqsubseteq Q$ are included in the search result by definition, the additional constraint $O_j \sqsubseteq Q$ of Lemma 15 in comparison to Lemma 1 causes no different usage of the lemma for the search algorithm. In other words, analogously to Lemma 1, Lemma 15 implies that

[3] Lemma 15 and 16 are the metric subset versions of Lemma 1 and 2, respectively.

the subset search algorithm can safely prune the covered subtree with root $T(O_r)$. The same holds true for Lemma 16 in comparison to Lemma 2.

Proof. Let O_j be an arbitrary but fixed object in the covered tree of O_r with $O_j \sqsubseteq Q$. The ordering $O_r \sqsubseteq O_j \sqsubseteq Q$ holds by definition of the SuperM-Tree's structure. Now, by definition of metric subset spaces, the triangle inequality

$$d(O_r, Q) \;\leqslant\; d(O_r, O_j) \;+\; d(O_j, Q)$$

holds. The structure of the SuperM-Tree requires $d(O_r, O_j) \leqslant r(O_r)$; thus,

$$\begin{aligned} d(O_r, Q) \;&\leqslant r(O_r) \;+\; d(O_j, Q) \quad \text{and} \\ d(O_j, Q) \;&\geqslant d(O_r, Q) - r(O_r). \end{aligned}$$

Together with the prerequisite $d(O_r, Q) > r(Q) + r(O_r)$ of the lemma, the desired inequality

$$d(O_j, Q) \;>\; r(Q) \;+\; r(O_r) \;-\; r(O_r) \;=\; r(Q)$$

holds. □

Lemma 16. If $d\big(P(O_r), Q\big) \;>\; r(Q) + r(O_r) + d\big(P(O_r), O_r\big)$, then $d(O_j, Q) > r(Q)$ holds for each object O_j with $O_j \sqsubseteq Q$ in the covered subtree of O_r.

Proof. Let O_j be an arbitrary but fixed object in the covered tree of O_r with $O_j \sqsubseteq Q$. The ordering $P(O_r) \sqsubseteq O_r \sqsubseteq O_j \sqsubseteq Q$ holds by definition of the structure of a SuperM-Tree. Hence, the triangle inequality yields

$$d\big(P(O_r), Q\big) \;\leqslant\; d\big(P(O_r), O_r\big) \;+\; d(O_r, O_j) \;+\; d(O_j, Q).$$

Now, since $d(O_r, O_j) \leqslant r(O_r)$ holds by definition and $d\big(P(O_r), Q\big) > r(Q) + r(O_r) + d\big(P(O_r), O_r\big)$ is a prerequisite, the inequality

$$r(Q) \;+\; r(O_r) \;+\; d\big(P(O_r), O_r\big) \;<\; d\big(P(O_r), O_r\big) \;+\; r(O_r) \;+\; d(O_j, Q)$$

holds; thus, $r(Q) < d(O_j, Q)$. □

5.3.3 Building the SuperM-Tree

The SuperM-Tree is a generalized search tree (GiST [46]), i. e., algorithms for insertion and deletion of objects manage an overflow or underflow of the nodes using split and merge operations, respectively.

The `insert` algorithm recursively descends the SuperM-Tree down to a leaf node that receives the new object. At each inner node, it follows the routing object with the smallest distance to the new object. Here, two events may occur:

1. Inserting the new object below the chosen routing object violates the total preorder condition of the SuperM-Tree. In this case, an exchange of the routing object is necessary.

2. After inserting the new object below the chosen routing object (or in the current leaf node), the node is overfilled. This case triggers a split of the node.

Also, at each routing object of the insertion path, the `insert` algorithm needs to assure that the covering radius $r(O_r)$ covers the newly inserted object. It does so by extending the covering radius as much as necessary. Algorithm 14 provides the pseudo-code for the `insert` algorithm.

Algorithm 14 SuperM-Tree Insert

```
1  Algorithm: supermtree.insert
2  Input: node N (default: root node), new object O*
3  if N is a leaf node
4    N := N ∪ {O*} // insert O* to N
5  else
6    O_r := arg min ( {d(O_r, O*) | O_r ⊑ O*} ∪ {d(O*, O_r) | O* ⊑ O_r} )
7    childSplit, O_1, O_2 :=insert(T(O_r), O*)
8    if childSplit
9      N := N \ {O_r} ∪ {O_1, O_2}
10   else if not O_r ⊑ O* // preorder condition violated
11     T(O*) := T(O_r) // configure new routing object
12     r(O*) := max_{O_c ∈ T(O*)} {d(O*, O_c) + r(O_c)}
13     N := N \ {O_r} ∪ {O*} // exchange routing object
14 didSplit, O_1, O_2 := split(N) // cf. Algorithm 15
15 if didSplit and N is root node
16   set new root node {O_1, O_2}
17 return didSplit, O_1, O_2
```

Split Management

As any other GiST tree, the SuperM-Tree provides a split strategy consisting of a `promotion` and a `partition` algorithm (cf. Algorithm 15). Splitting a node N makes it two nodes N_1 and N_2, each getting a new parent routing object.

The `promote` algorithm provides the two routing objects that ought to replace the old routing object in the parent node. If N is the root node, a new node filled with the two promoted routing objects will serve as the tree's new root node.

The `partition` algorithm distributes the objects of the node N among the two new nodes N_1 and N_2. The algorithm in this thesis follows the *generalized hyperplane* strategy that puts each object to its nearest promoted routing object (cf. Algorithm 16).

Algorithm 15 SuperM-Tree Split

```
1  Algorithm: supermtree.split
2  Input: node N, object O
3  if #N ⩽ capacity
4    return false, nil, nil
5  foundPromotion, O1, O2 ≔ promote(N) // cf. Algorithm 17
6  if not foundPromotion:
7      return false, nil, nil
8  N1, N2 ≔ partition(N, O1, O2) // cf. Algorithm 16
9  T(O1) ≔ N1
10 T(O2) ≔ N2
11 r(O1) ≔ max{d(O1, O) | O ∈ N1}
12 r(O2) ≔ max{d(O2, O) | O ∈ N2}
13 return true, O1, O2
```

A particular combination of a `promote` and `partition` strategy is called a *split policy*. This thesis only discusses the split policy presented by the following sections since it suffices for the SuperM-Tree's proof of concept.

Algorithm 16 SuperM-Tree Partition

```
1  Algorithm: supermtree.partition
2  Input: node N, routing objects O1, O2
3  N1 ≔ {Oj ∈ N | d(O1, Oj) < d(O2, Oj) ∧ O1 ⊑ Oj}
4  N2 ≔ N \ N1
5  return N1, N2
```

Promotion Strategy

This subsection explains the details of the promotion strategy used in this thesis (cf. Algorithm 17). Although the implementation calls the `partition` algorithm multiple times in the pseudo-code, the final implementation actually merges both the `promote` and `partition` function to improve runtime performance.

The motivation for the choice of the promotion strategy is based on the following fact: Minimizing the overlap of covering areas in a node turned out to be crucial for the performance of multi-dimensional index structures,

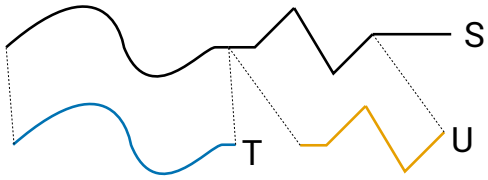


Figure 5.4: Example for two very distinct time series (T and U) being similar to a third one (S).

including the R*-Tree [16], and the M-Tree [35]. Considering a node N with two routing objects S and T in an M-Tree, the distance between both covering areas is $\psi := d(S,T) - r(S) - r(T)$. Both covering areas overlap when this value ψ becomes negative. The overlap gets larger the more this value ψ shrinks below zero. Furthermore, the probability increases that a query overlaps both covering areas. Hence, maximizing ψ decreases the probability of having to descend multiple paths.

Since the concept of the *volume of an overlap* does not exist generally for metric spaces, the negative of this value ψ is considered as overlap, i. e., $\omega(S,T) := r(S) + r(T) - d(S,T)$.

Virtual distance and overlap: The following situation shows that the concept of the overlap as described above does not fit in metric subset spaces: Consider the time series S, T, and U that are sketched in Figure 5.4, and consider T and U to be actual routing objects, one of which covers S. While T and U are not similar at all (and thus might yield no overlap for sufficiently small radii), S has a small distance to both objects T and U, i. e., a query for a time series Q *containing* S[4] would need to traverse both paths through the routing objects T and U. To avoid such situations, the concept of *overlap* needs to consider the objects in the covered subtrees of the routing objects.

This example motivates introducing the concept of *virtual distances* regarding a set of third party objects.

Definition 15. Let $\mathcal{R} \subset \mathcal{M}$ be a set of metric subset objects, and let $S, T \in \mathcal{M}$. Then,

$$v_{\mathcal{R}}(S,T) := \min \Big\{ d(T,R) + d(S,R) \;\mid\; R \in R, \quad S \sqsubseteq R, \quad T \sqsubseteq R \Big\}$$

is called the *virtual distance* of S and T regarding $\mathcal{R}$.

Consider a node with routing objects $\mathcal{R}$ and two arbitrary objects S and T. If the virtual distance $v_{\mathcal{R}}(S,T)$ is small then there is an $R \in \mathcal{R}$ with $S, T \sqsubseteq R$ such that both distances $d(T,R)$ and $d(S,R)$ are small. This

[4] Here, Q "containing" S means that $S \sqsubseteq Q$ and $d(S,Q)$ is small.

routing object R probably has a child C with $R \sqsubseteq C$ such that $d(R, C)$ is small.[5] The triangle inequality yields that $d(S, C) \leqslant d(S, R) + d(R, C)$ and $d(T, C) \leqslant d(T, R) + d(R, C)$, i. e., $d(S, C)$ and $d(T, C)$ are small too. If S and T ought to be promoted to become routing objects in the same node as R then a query that matches C would probably need to search the subtrees for S and T too. Hence, the promotion of S and T increases the computation time of range queries, which makes them a bad choice for promotion.

On the other hand, if the virtual distance $v_{\mathcal{R}}(S, T)$ is large then at least one of $d(T, R)$ and $d(S, R)$ for each $R \in \mathcal{R}$ with $S, T \sqsubseteq R$ is large. Assume that S and T are promoted as routing objects in the same node as $\mathcal{R}$. Then, a query for an object Q is less likely to descend the tree through both S and T if the query matches an object below any of the $R \in \mathcal{R}$ with $S, T \sqsubseteq R$. Therefore, this section proposes a promotion strategy that promotes a pair of objects minimizing the *virtual overlap* $r(S) + r(T) - v_{\mathcal{R}}(S, T)$ where $r(S)$ and $r(T)$ are the radii after partitioning is the set of (routing) objects within the node.

Definition 16. For two routing objects S, T and $\mathcal{R} \subset \mathcal{M}$, the *virtual overlap of S and T regarding $\mathcal{R}$* is

$$\omega_{\mathcal{R}}(S, T) \;\coloneqq\; r(S) + r(T) - v_{\mathcal{R}}(S, T).$$

Strict order in insertion paths: The SuperM-Tree maintains the strict order $P \sqsubseteq T$ for objects T with parent P. To keep this property, the promotion strategy chooses two of the smallest objects of a node (cf. Line 3–5 in Algorithm 17).

Partition Strategy

M-Trees that are built with the geometric approach (cf. *generalized hyperplane* in [35]) are superior in query performance compared to trees built with the balanced strategy (i. e., splitting to partitions of equal size). Since the SuperM-Tree is derived from the M-Tree, this insight is assumed to hold for the SuperM-Tree as well. For this reason, the algorithm in this thesis implements the geometric approach.

However, during the development process of the SuperM-Tree, the following effect occurred: Given a small object (e. g., a time series of length 1) and a larger object, the former has a lower expected distance to a third object than the latter. When promoting two objects, one of them is usually smaller than the other. Hence, most other node objects are more similar to the smaller object, and the partitions become more unbalanced after multiple insertions. At some point, partitions even degraded such that the

[5] This statement assumes that routing objects are (nearly) centered representatives of clusters.

Algorithm 17 SuperM-Tree Promote Algorithm

```
1  Algorithm: supermtree.promote
2  Input: node N
3  C ≔ {O | ∀O_i ∈ N : O ⊑ O_i}
4  if #C = 1 // make sure #C ⩾ 2
5    C ≔ C ∪ {O | ∀O_j ∈ N \ C : O ⊑ O_j}
6  e ≔ ∞
7  P_1 ≔ nil
8  P_2 ≔ nil
9  foundPromotion ≔ false
10 for each pair O_1, O_2 ∈ C
11   N_1, N_2 ≔ partition(N, O_1, O_2) // cf. Algorithm 16
12   if overfilled nodes allowed and (#N_1 ⩽ 1 or #N_2 ⩽ 1)
13     skip // ignore O_1, O_2
14   else if ω_N(O_1, O_2) < e
15     foundPromotion ≔ true
16     P_1 ≔ O_1
17     P_2 ≔ O_2
18     e ≔ ω_N(O_1, O_2)
19 return foundPromotion, P_1, P_2
```

split algorithm only moves one object (the larger promoted object) out of the node. The resulting tree degraded to a large trunk with lots of thin branches, often holding only one object. This structure results in slow query performance since pruning thin branches converges to a linear scan search algorithm's behavior.

Ignoring the maximum capacity of a node prevents the behavior described above: Instead of strictly splitting nodes when they exceed the capacity, a split is realized if the proposed partitions are not degenerated (cf. Line 12 in Algorithm 17). Section 5.4 shows that this strategy outperforms the strict split execution.

5.4 Evaluation

This section provides an evaluation of the runtime when building and when querying the SuperM-Tree.

A second goal is to evaluate the flexibility of the concept of metric subset spaces. To this end, the evaluation consists of experiments on three modular metric subset distance functions: the Euclidean distance on subsequences, the Dog Keeper distance on subsequences, and the Hausdorff distance on subsets. For each application, the evaluation compares both split policies

described in Section 5.3.3 (i. e., fixed capacities; and allowing large nodes) against a linear scan search algorithm. The experiments executing a linear scan search algorithm speedup the runtime already by using lower bounds to the considered distance function (cf. [10]).

Runtime comparisons: To understand the influence of the dataset's different properties, most experiments are based on synthetic datasets. The sequence-based distance functions (δ_2 and `S-DK`) are applied on random sequences of uniformly distributed lengths between 1 and 128. The set-based distance function (HAUSDORFF) is applied on random sets of uniformly distributed cardinality between 1 and 32. The elements of all sequences and sets are uniformly distributed over a fixed finite interval. The evaluation of the runtime's dependency on the dataset's size conducts experiments on datasets with 2^8 to 2^{23} elements.

The node's capacity is set to 128.[6]. Each experiment averages the runtime over 100 queries. Figure 5.5 shows the runtime for building the tree with successive insert operations, and Figure 5.6 shows the average query times on the resulting trees.

To compare the results with real-world examples, the evaluation conducts the same experiments for both sequence-based applications on the UCR time series benchmark suite [67]. The UCR time series suite consists of multiple datasets, each of which is split into a training and a test set. A tree is built for both sequence-based distance functions from each training set of these datasets, respectively. For each tree, the query runtime is averaged over all samples from the corresponding test sets. Since the UCR suite is not designated for subsequence queries, the training sequences are cropped down to random subsequences with a uniformly distributed length between 1 and 128. Figure 5.7 shows that the speedup follows the same trend as in the experiments with the synthetic datasets.

Fixed capacity vs. large nodes: Figure 5.5 shows that the dynamic capacity split policy (*large nodes*) outperforms the static capacity split policy (*fixed capacity*) by more than one order of magnitude while building the tree. During the experiments with the fixed capacity strategy, keeping track of the node's size revealed a drastic fall down of the sizes towards an average of approximately one for small datasets already (i. e., a few thousand objects). This observation confirmed the assumption that the tree degenerates as explained in Section 5.3.3.

[6]During the development process of the SuperM-Tree, the node's capacity had no significant impact on the performance, i. e., runtimes were stable for node capacities from 64 to 512.

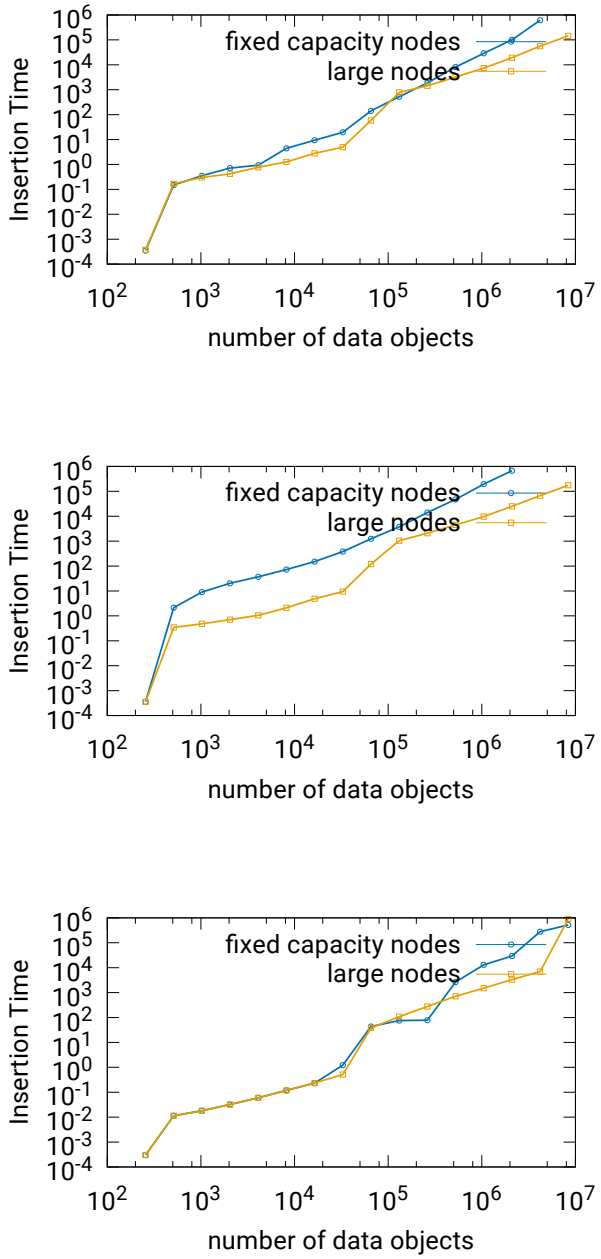


Figure 5.5: SuperM-Tree: Building time with synthetic data in seconds (top: δ_2; middle: `S-DK`; bottom: `S-HD`).

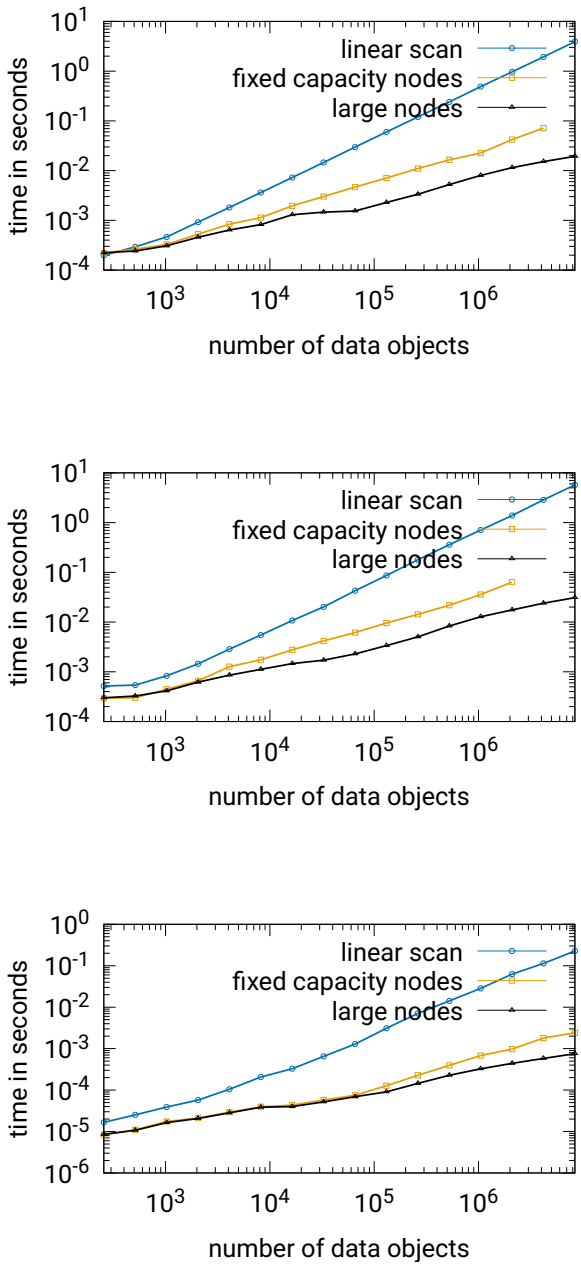


Figure 5.6: SuperM-Tree: Average query time of 100 1-NN queries on synthetic data(top: δ_2; middle: `S-DK`; bottom: `S-HD`).

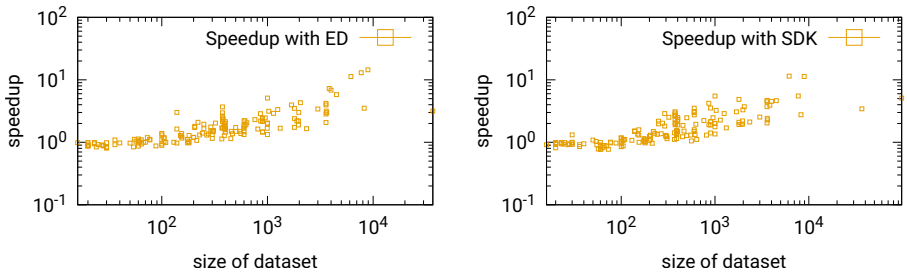


Figure 5.7: SuperM-Tree: Average Speedup against linear scan on UCR datasets with L2 (top) and `S-DK` (bottom)

The effect of dimensionality: The *curse of dimensionality* is a well-studied effect and occurs in the SuperM-Tree as well. The effect appears as seen in other multi-dimensional index structures. Since there is no new insight in this, this section skips presenting experimental results on this topic.

Variance of the object's sizes: As mentioned earlier, the sequences are cropped down to subsequences of random lengths. Earlier experiments showed that this is a crucial step for gaining speedup.[7] Hence, the performance strongly depends on having small elements in the dataset.

While splitting overfilled nodes, the smaller objects are being promoted as routing objects. Intuitively, these objects form a space with lower (intrinsic) dimensionality in the higher levels of the tree (i. e., closer to the root).[8] Thus, the curse of dimensionality is alleviated in the higher levels; the pruning works better in the higher levels; and the overall number of pruned objects increases. On the other hand, datasets with only large objects form a high dimensional space in the higher levels already, and thus the curse of dimensionality affects the query runtime.

Extending the metric subset space with a function generating smaller (promotion) objects from a given set of objects might solve that problem. Alternatively, small random elements could be inserted as meta elements manually before or during the whole insertion process. However, this opens new research areas for each metric subset space, respectively.

[7] Detailed information is not included here since it gains no more insight.

[8] For an example, consider the Euclidean distance on real-valued time series of equal length: There, the length of the time series defines the dimensionality of the time series space.

5.5 Conclusion

This section introduced metric subset spaces as a semantic extension of metric spaces. Three applications from different fields show the flexibility of this concept.

This section further introduced the SuperM-Tree, a data structure for metric subset spaces derived from the M-Tree. The SuperM-Tree provides nearest neighbor queries searching for *subobjects* (e. g., subsequences or subsets). The experiments show that the index structure outperforms the linear scan by multiple orders of magnitude on large datasets. However, the experiments also revealed that small objects are necessary in the dataset since they act as low dimensional routing objects, and therefore alleviate the curse of dimensionality. As future work, these small objects could be inserted as meta elements manually or generated in a new promotion strategy depending on the specific metric subset space.

Chapter 6

Summary

This chapter summarizes the results of this thesis in Section 6.1. Section 6.2 discusses the results, including possible approaches for improving them. Finally, Section 6.3 concludes this thesis.

6.1 Summary of Results

This thesis categorized well-established distance functions regarding two semantic properties: Do they consider time warping? Do they consider congruence (isometric) transformations? Both properties have been axiomatized formally, and the runtime of corresponding metric distance functions has been analyzed theoretically.

For the Dog Keeper distance, this thesis provides a new algorithm and shows that it outperforms a canonical extension of $\texttt{LB}_{\texttt{Keogh}}$ for `DTW` in high dimensional spaces. This thesis proves that, unfortunately, there is no fast algorithm computing the congruence distance unless P=NP. On the other hand, this thesis provides fast metric distance functions tightly approximating the congruence distance.

Furthermore, this thesis enhances time-warping and congruence metric distance functions by introducing the novel concept of metric subset spaces and metric subset distance functions to support subsequence search. As a proof of concept, this thesis presents the SuperM-Tree that is an index structure similar to metric index structures but indexing metric subset spaces. Similar to metric index structures, it provides nearest neighbor queries on the indexed datasets. The experiments show that it outperforms fast linear scan algorithms, especially on large datasets (millions of elements).

6.2 Discussion

This section discusses various benefits, drawbacks, interesting side-effects, and possible future work of the results summarized in Section 6.1.

Time-warping – robustness vs. computation time: The comparison of DTW and DK in Section 3.5 showed that both yield similar accuracy on synthetic and real-world datasets.

However, DK is more vulnerable to noisy data while DTW with $\mathtt{LB}_{\mathtt{Keogh}}$[1] is vulnerable to datasets with larger temporal distortion. The reason for DTW being more robust against noisy data is that it computes the L_1-norm along a warping path, and noisy entries or outliers drown in the sum of all small distance values. The DK distance computes the L_∞-norm along a warping path, and outliers dominate the L_∞-norm no matter how long the warping path is.

On the other hand, DK is much faster to compute because early abandoning works well on the L_∞-norm: While the computation of DTW exceeds a given threshold more likely in the later computation steps, the computation of DK does that more likely in the early computation steps.

The insights mentioned above yield to use the L_p norm for some $p \in \mathbb{R}$ with $1 \leqslant p \leqslant \infty$. This parameter would enable a trade-off between computation speed and robustness against noise: For values closer to 1, the computation is slower but more robust; for values closer to ∞, the computation is faster but less robust against noise. However, any value smaller than ∞ will violate the triangle inequality of the DK distance and hence prohibit the usage of metric index structures.

Time-warping – triangle inequality vs. Sakoe-Chiba band: The restriction of temporal distortion such as the Sakoe-Chiba band or the Itakura parallelogram is also applicable to the DK distance. On the other hand, the application of this restriction violates the triangle inequality of the DK distance.

It is possible, however, to apply the Sakoe-Chiba band to the DK distance such that the triangle inequality holds under certain circumstances, i. e., when the bandwidths are adapted accordingly. For example, the triangle inequality $\mathtt{DK}(A, C) \leqslant \mathtt{DK}(A, B) + \mathtt{DK}(B, C)$ holds for time series A, B, and C if the bandwidth in $\mathtt{DK}(A, C)$ is the sum of the bandwidths in $\mathtt{DK}(A, B)$ and $\mathtt{DK}(B, C)$. The downside of such an approach is that index structures for metric (subset) spaces need adaption to comply with this limitation, i. e., they are not as generic anymore but rather fixed to this specific distance function.

[1] Considering pure DTW without any lower bounds is not practical because of its slow runtime.

Congruence – strength of theory: The main contribution of Chapter 4 was the proof of Theorem 3 that claims that the approximated congruence problem is hard to compute unless P=NP. A close study of the proof reveals that the theorem is true even if only rotation and reflection of the time series were allowed. Still, the proof only works for the congruence distance based on the L_1-norm. Can this proof be extended to work on any other norm?

Congruence – application on other domains: An interesting application for the congruence distance (or at least one of its approximations) is the distances of subgraphs in huge graphs, e. g., the web or tweets on Twitter[2]. In these examples, a time series can specify an ordered list of web pages visited by a user or the ordered list of channels on that a tweet was retweeted, respectively. Congruent time series could indicate similar browsing or broadcasting behavior. Finding such similar time series is easy if they are considered as congruent time series. In this case, the (fast) delta distance describes a way to construct a hash value for each time series. Congruent time series (i. e., similar browsing or broadcasting behavior) are then easy to find using index structures for multi-dimensional data. Behavior analysis would be possible by clustering these hash values.

Congruence – discussion on approximations: The (fast) delta distance seems like an ad-hoc solution for an approximation to the congruence distance. Indeed, the delta distance's purpose is to show the basic idea for how to approximate the congruence distance. The greedy delta distance uses this idea and improves it already. On the other hand, the greedy delta distance does not satisfy the triangle inequality anymore and is therefore not applicable for metric index structures. Given a particular dataset, a statistical analysis of the indices picked by the greedy delta distance might provide a static set of indices that improve the tightness over that of the (fast) delta distance but still preserving the metric properties.

Congruence aware time-warping distance functions: In the end, robustness against time-distortion is indispensable for distance functions on time series. It is probably even more essential than being invariant under isometric transformations (i. e., congruence distance functions). Still, the question arises whether time-warping is applicable to the congruence distance, e. g., can time-warping be applied on the self-similarity matrices of time series? That approach would yield a time-warping aware congruence measure. While this is canonically possible by interpreting the columns of the self-similarity matrix as elements in a time series, the implementation is not feasible with dynamic programming algorithms as they compute `DTW` or `DK`. Is there a better way to solve this issue?

[2]Twitter is a communication platform that allows users to *tweet* short messages, follow the channels of other users, and *retweet* other users' messages on their own channel.

SuperM-Tree – discussion on metric subset spaces: Metric subset spaces seem to be a promising extension of metric spaces to support nearest neighbor queries on sized objects. It is unclear at this point what mathematical results can be expected on metric subset spaces. While the total preorder of metric subset spaces describes the size of objects, it leaves open whether the different equivalence classes regarding the size correspond to natural numbers, real numbers, or even something else. Considering the distance functions, an intuitive comparison of metric spaces and metric subset spaces sketches metric spaces as flat areas and metric subset spaces as hierarchies.

SuperM-Tree – subset vs. superset queries: The SuperM-Tree seems to work better if the top-level nodes contain *small* objects that yield a small intrinsic dimensionality [31]. Although Corollary 3 claims that superset queries are technically possible, this insight reveals that such trees would not perform well because they would have large objects inducing a high intrinsic dimensionality in the top-level nodes. Hence, the SuperM-Tree should rather be used for subset queries, i. e., queries that search for similar (labeled) *snippets* of the query in the dataset.

SuperM-Tree – improving query runtime: For datasets that do not contain small elements inducing low intrinsic dimensionality in the higher levels of the tree, the SuperM-Tree performs worse, which is a crucial handicap. Additionally to the distance function, a function that produces small objects out of large ones might solve this issue. The SuperM-Tree could use such a *chopping* function in the promotion algorithm to promote smaller objects to the tree's higher levels. Nevertheless, this moves the SuperM-Tree further away from the generic solution that it is supposed to be. Although the SuperM-Tree provides a hint for a possible direction of a generic, efficient index structure for subsequence search, it remains an open challenge to find such.

6.3 Conclusion

This thesis aims to provide new methods for nearest neighbor search algorithms that are efficient on metric time series, in particular high-dimensional time series. Regarding time-warping distance functions, Chapter 3 achieved this goal by pointing towards the `DK` distance that is superior to `DTW` on high-dimensional time series. Chapter 4 showed that measuring the congruence of time series in Euclidean spaces is a difficult problem; thus, approximations are essential for practical applications.[3] Moreover, Chapter 4 provides a template for constructing such approximations. In the end, Chapter 5 provides the concept of metric subset spaces, a new basis for building generic index structures for a large variety of nearest neighbor queries asking for subsets. The solutions proposed in the previous chapters are examples of distance functions that are compatible with this new concept. Hence, this thesis opens new opportunities for an established topic that still remains challenging.

[3] On the other hand, the existence of a polynomial-time algorithm that computes the congruence of two time series up to a certain precision proves P=NP.

Bibliography

[1] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient similarity search in sequence databases. In *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*, FODO '93, page 69–84, Berlin, Heidelberg, 1993. Springer-Verlag.

[2] T. Akutsu. On determining the congruence of point sets in d dimensions. *Computational Geometry*, 9(4):247 – 256, 1998.

[3] G. Al-Naymat, S. Chawla, and J. Taheri. Sparsedtw: A novel approach to speed up dynamic time warping. In *Proceedings of the Eighth Australasian Data Mining Conference - Volume 101*, AusDM '09, page 117–127, AUS, 2009. Australian Computer Society, Inc.

[4] H. Alt and L. J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In J. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 121–153. North Holland / Elsevier, 2000.

[5] H. Alt, K. Mehlhorn, H. Wagener, and E. Welzl. Congruence, similarity, and symmetries of geometric objects. *Discrete & Computational Geometry*, 3(3):237–256, Jan. 1988.

[6] E. M. Arkin, K. Kedem, J. S. B. Mitchell, J. Sprinzak, and M. Werman. Matching Points into Pairwise-Disjoint Noise Regions: Combinatorial Bounds and Algorithms. *INFORMS Journal on Computing*, 4(4):375–386, November 1992.

[7] M. D. Atkinson. An optimal algorithm for geometrical congruence. *Journal of Algorithms*, 8(2):159–172, June 1987.

[8] J. P. Bachmann. The SuperM-Tree: Indexing metric spaces with sized objects. *CoRR*, abs/1901.11453, 2019.

[9] J. P. Bachmann and J. Freytag. Dynamic time warping and the (windowed) dog-keeper distance. In C. Beecks, F. Borutta, P. Kröger, and T. Seidl, editors, *Similarity Search and Applications - 10th International Conference, SISAP 2017, Munich, Germany, October 4-6, 2017,*

Proceedings, volume 10609 of *Lecture Notes in Computer Science*, pages 127–140. Springer, 2017.

[10] J. P. Bachmann and J. Freytag. An analytical approach to improving time warping on multidimensional time series. *CoRR*, abs/1811.11076, 2018.

[11] J. P. Bachmann and J. Freytag. Efficient measuring of congruence on high dimensional time series. *CoRR*, abs/1811.11856, 2018.

[12] J. P. Bachmann and J. Freytag. High dimensional time series generators. *CoRR*, abs/1804.06352, 2018.

[13] J. P. Bachmann, J. Freytag, B. Hauskeller, and N. Schweikardt. Measuring congruence on high dimensional time series. *CoRR*, abs/1805.10697, 2018.

[14] J. P. Bachmann, K. M. Trogant, and J.-C. Freytag. GPU-beschleunigte time warping-distanzen. *PARS-Mitteilungen*, 35(1):63–72, 2020.

[15] A. Backurs and P. Indyk. Edit distance cannot be computed in strongly subquadratic time (unless seth is false). In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, page 51–58, New York, NY, USA, 2015. Association for Computing Machinery.

[16] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. *ACM SIGMOD Record*, 19(2):322–331, May 1990.

[17] S. Berchtold, D. A. Keim, and H.-P. Kriegel. The X-Tree: An index structure for high-dimensional data. In T. M. Vijayaraman, editor, *Proceedings of the Twenty-second International Conference on Very Large Data-Bases; Mumbai (Bombay), India 3 - 6 September, 1996*, pages 28–39, San Francisco, 1996. Morgan Kaufmann.

[18] A. C. Berry. The accuracy of the gaussian approximation to the sum of independent variates. *Transactions of the American Mathematical Society*, 49(1):122–136, 1941.

[19] A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 97–104, New York, NY, USA, 2006. ACM.

[20] K. Bhaduri, Q. Zhu, N. C. Oza, and A. N. Srivastava. Fast and flexible multivariate time series subsequence search. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, page 48–57, USA, 2010. IEEE Computer Society.

[21] V. Bijalwan, P. Kumari, J. P. Espada, and V. B. Semwal. Machine learning approach for text and document mining. *CoRR*, abs/1406.1580, 2014.

[22] E. G. Birgin and J. M. Martínez. Improving ultimate convergence of an augmented lagrangian method. *Optimization Methods Software*, 23(2):177–195, Apr. 2008.

[23] T. Bozkaya and M. Ozsoyoglu. Distance-based indexing for high-dimensional metric spaces. *ACM SIGMOD Record*, 26(2):357–368, June 1997.

[24] M. Brill, T. Fluschnik, V. Froese, B. Jain, R. Niedermeier, and D. Schultz. Exact mean computation in dynamic time warping spaces. *Data Mining and Knowledge Discovery*, 33(1):252–291, 2019.

[25] K. Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless seth fails. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 661–670, 2014.

[26] K. Bringmann and M. Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. *IEEE Symposium on Foundations of Computer Science*, pages 79–97, 2015.

[27] S. Cabello, P. Giannopoulos, and C. Knauer. On the parameterized complexity of d-dimensional point set pattern matching. *Information Processing Letters*, 105(2):73–77, Jan. 2008.

[28] A. Camerra, T. Palpanas, J. Shieh, and E. Keogh. iSAX 2.0: Indexing and mining one billion time series. In *2010 IEEE International Conference on Data Mining*, ICDM '10, page 58–67, USA, 2010. IEEE Computer Society.

[29] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Transactions on Database Systems (TODS)*, 27(2):188–228, June 2002.

[30] K.-P. Chan and A. W.-C. Fu. Efficient time series matching by wavelets. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, page 126, Los Alamitos, CA, USA, Mar. 1999. IEEE Computer Society.

[31] E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, Sept. 2001.

[32] L. Chen. *Similarity Search over Time Series and Trajectory Data*. PhD thesis, Waterloo, Ont., Canada, 2005.

[33] L. Chen and R. Ng. On the marriage of lp-norms and edit distance. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, VLDB '04, pages 792–803. VLDB Endowment, 2004.

[34] Q. Chen, L. Chen, X. Lian, Y. Liu, and J. X. Yu. Indexable PLA for efficient similarity search. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, page 435–446. VLDB Endowment, 2007.

[35] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, VLDB '97, pages 426–435, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[36] D. Comer. Ubiquitous B-tree. *ACM Computing Surveys*, 11(2):121–137, June 1979.

[37] A. Degani, M. Dalai, R. Leonardi, and P. Migliorati. A heuristic for distance fusion in cover song identification. In *2013 14th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, pages 1–4, 2013.

[38] M. M. Deza and E. Deza. *Encyclopedia of Distances*. Springer Berlin Heidelberg, 2013.

[39] J.-P. Eckmann, S. O. Kamphorst, and D. Ruelle. Recurrence plots of dynamical systems. *Europhysics Letters (EPL)*, 4(9):973–977, Nov. 1987.

[40] T. Eiter and H. Mannila. Computing discrete fréchet distance. Technical report, Technische Universität Wien, 1994.

[41] P. Esling and C. Agon. Time-series data mining. *ACM Computing Surveys*, 45(1):12:1–12:34, Dec. 2012.

[42] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. *ACM SIGMOD Record*, 23(2):419–429, May 1994.

[43] M. R. Fréchet. *Sur quelques points du calcul fonctionnel*. 22. Rendiconti del Circolo Mathematico di Palermo, 1906.

[44] A. W.-C. Fu, E. Keogh, L. Y. Lau, C. A. Ratanamahatana, and R. C.-W. Wong. Scaling and time warping in time series querying. *The VLDB Journal*, 17(4):899–921, July 2008.

[45] P. J. Heffernan and S. Schirra. Approximate decision algorithms for point set congruence. *Computational Geometry*, 4(3):137–156, 1994.

[46] J. M. Hellerstein, J. F. Naughton, and A. Pfeffer. Generalized search trees for database systems. In *Proceedings of the 21th International Conference on Very Large Data Bases*, VLDB '95, pages 562–573, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[47] H. Herrlich, H. Bargenda, and C. Trompelt. *Einführung in die Topologie*. Berliner Studienreihe zur Mathematik. Heldermann Verlag, 1986.

[48] G. R. Hjaltason and H. Samet. Index-driven similarity search in metric spaces (survey article). *ACM Transactions on Database Systems*, 28(4):517–580, Dec. 2003.

[49] D. P. Huttenlocher, G. A. Klanderman, and W. A. Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, Sept. 1993.

[50] P. Indyk and S. Venkatasubramanian. Approximate congruence in nearly linear time. *Computational Geometry*, 24(2):115–128, Feb. 2003.

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

[52] M. J. D. Powell. The BOBYQA algorithm for bound constrained optimization without derivatives. Technical report, Centre for Mathematical Sciences, University of Cambridge, UK, 2009. Report No. DAMTP 2009/NA06.

[53] R. Jampani and V. Pudi. Using Prefix-Trees for efficiently computing set joins. In L. Zhou, B. C. Ooi, and X. Meng, editors, *Database Systems for Advanced Applications*, page 761–772, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[54] J. H. Jensen, M. G. Christensen, D. P. W. Ellis, and S. H. Jensen. A tempo-insensitive distance measure for cover song identification based on chroma features. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2209–2212, 2008.

[55] S. G. Johnson. The NLopt nonlinear-optimization package. `http://github.com/stevengj/nlopt`.

[56] M. W. Kadous. *Temporal Classification: Extending the Classification Paradigm to Multivariate Time Series*. PhD thesis, University of New South Wales, Australia, 2002.

[57] E. Keogh, T. Palpanas, V. B. Zordan, D. Gunopulos, and M. Cardle. Indexing large human-motion databases. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, page 780–791. VLDB Endowment, 2004.

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

[59] E. Keogh, L. Wei, X. Xi, M. Vlachos, S.-H. Lee, and P. Protopapas. Supporting exact indexing of arbitrarily rotated shapes and periodic time series under euclidean and warping distance measures. *The VLDB Journal*, 18(3):611–630, 2009.

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

[61] F. Kin-Pong Chan, A. Wai-chee Fu, and C. Yu. Haar wavelets for efficient similarity search of time-series: With and without time warping. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):686–705, Mar. 2003.

[62] F. Korn, H. V. Jagadish, and C. Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. *ACM SIGMOD Record*, 26(2):289–300, June 1997.

[63] A. Kunkel, A. Rheinländer, C. Schiefer, S. Helmer, P. Bouros, and U. Leser. Piejoin: Towards parallel set containment joins. In *Proceedings of the 28th International Conference on Scientific and Statistical Database Management*, SSDBM '16, pages 11:1–11:12, New York, NY, USA, 2016. ACM.

[64] D. Lemire. Faster retrieval with a two-pass dynamic-time-warping lower bound. *Pattern Recognition*, 42(9):2169–2180, Sept. 2009.

[65] V. Levenshtein. Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information Transmission*, 1:8–17, 1965.

[66] S. Lian, N. Nikolaidis, and H. T. Sencar. Content-based video copy detection – a survey. In H. T. Sencar, S. Velastin, N. Nikolaidis, and

S. Lian, editors, *Intelligent Multimedia Analysis for Security Applications*, pages 253–273. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[67] M. Lichman. UCI machine learning repository, University of California, Irvine, School of Information and Computer Sciences. `http://archive.ics.uci.edu/ml`, 2013.

[68] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, DMKD '03, page 2–11, New York, NY, USA, 2003. Association for Computing Machinery.

[69] J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing SAX: A novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15(2):107–144, Oct. 2007.

[70] Y. Liu, D. Zhang, G. Lu, and W.-Y. Ma. A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40(1):262–282, Jan. 2007.

[71] T. M. Rath and R. Manmatha. Lower-bounding of dynamic time warping distances for multivariate time series. `https://works.bepress.com/r_manmatha/19/`, Feb. 2003.

[72] S. Mitra and T. Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(3):311–324, May 2007.

[73] M. Müller. *Information Retrieval for Music and Motion*. Springer-Verlag, Berlin, Heidelberg, 2007.

[74] R. Neamtu, R. Ahsan, E. Rundensteiner, and G. Sarkozy. Interactive time series exploration powered by the marriage of similarity distances. *Proceedings of the VLDB Endowment*, 10(3):169–180, Nov. 2016.

[75] F. Nigsch, A. Bender, B. van Buuren, J. Tissen, E. Nigsch, and J. B. O. Mitchell. Melting point prediction employing k-nearest neighbor algorithms and genetic parameter optimization. *Journal of Chemical Information and Modeling*, 46(6):2412–2422, 2006.

[76] D. Novak and M. Batko. Metric index: An efficient and scalable solution for similarity search. In *2009 Second International Workshop on Similarity Search and Applications*, pages 65–73, Aug 2009.

[77] F. Palumbo, C. Gallicchio, R. Pucci, and A. Micheli. Human activity recognition using multisensor data fusion based on reservoir computing.

Journal of Ambient Intelligence and Smart Environments, 8:87–107, 2016.

[78] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 262–270, New York, NY, USA, 2012. ACM.

[79] N. Saito. *LOCAL FEATURE EXTRACTION AND ITS APPLICATIONS USING A LIBRARY OF BASES*. PhD thesis, Department of Mathematics, Yale University, 2011.

[80] H. Sakoe and S. Chiba. Readings in speech recognition. In A. Waibel and K.-F. Lee, editors, *Readings in Speech Recognition*, chapter Dynamic Programming Algorithm Optimization for Spoken Word Recognition, pages 159–165. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[81] P. Schäfer and M. Högqvist. SFA: A symbolic fourier approximation and index for similarity search in high dimensional datasets. In *Proceedings of the 15th International Conference on Extending Database Technology*, EDBT '12, page 516–527, New York, NY, USA, 2012. Association for Computing Machinery.

[82] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh. Indexing multi-dimensional time-series with support for multiple distance measures. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 216–225, New York, USA, 2003. ACM.

[83] R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, Jan. 1974.

[84] S. Wandelt, J. Starlinger, M. Bux, and U. Leser. Rcsi: Scalable similarity search in thousand(s) of genomes. *Proceedings of the VLDB Endowment*, 6(13):1534–1545, Aug. 2013.

[85] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309, Mar. 2013.

[86] A. Wary and A. Neelima. A review on robust video copy detection. *International Journal of Multimedia Information Retrieval*, 8(2):61–78, Aug. 2019.

[87] J. Yang, W. Zhang, S. Yang, Y. Zhang, X. Lin, and L. Yuan. Efficient set containment join. *The VLDB Journal*, 27(4):471–495, Aug 2018.

[88] B.-K. Yi, H. V. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *Proceedings of the Fourteenth International Conference on Data Engineering*, ICDE '98, page 201–208, USA, 1998. IEEE Computer Society.

[89] P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proc. of the ACM-SIAM Symposium on Discrete algorithms*, SODA, pages 311–321. Society for Industrial and Applied Mathematics, 1993.

[90] P. Zezula, M. Batko, and V. Dohnal. Indexing metric spaces. In L. LIU and M. T. ÖZSU, editors, *Encyclopedia of Database Systems*, pages 1451–1454. Springer US, Boston, MA, 2009.

Humboldt-Universität zu Berlin
Philosophische Fakultät II

Name: Schäfer Vorname: Jörg Peter
Matrikelnummer: 196605

Selbstständigkeitserklärung zur Dissertation

Ich erkläre ausdrücklich, dass es sich bei der von mir eingereichten schriftlichen Arbeit mit dem Titel

Efficient Nearest Neighbor Search on Metric Time Series

um eine von mir selbstständig und ohne fremde Hilfe verfasste Arbeit handelt.

Ich erkläre ausdrücklich, dass ich *sämtliche* in der oben genannten Arbeit verwendeten fremden Quellen, auch aus dem Internet (einschließlich Tabellen, Grafiken u. Ä.) als solche kenntlich gemacht habe. Insbesondere bestätige ich, dass ich ausnahmslos sowohl bei wörtlich übernommenen Aussagen bzw. unverändert übernommenen Tabellen, Grafiken u. Ä. (Zitaten) als auch bei in eigenen Worten wiedergegebenen Aussagen bzw. von mir abgewandelten Tabellen, Grafiken u. Ä. anderer Autorinnen und Autoren (Paraphrasen) die Quelle angegeben habe.

Mir ist bewusst, dass Verstöße gegen die Grundsätze der Selbstständigkeit als Täuschung betrachtet und entsprechend der Prüfungsordnung und/oder der Allgemeinen Satzung für Studien- und Prüfungsangelegenheiten der HU (ASSP) geahndet werden.

Datum 19.03.2021 Unterschrift