UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Large-scale similarity-based time series mining

Diego Furtado Silva

Tese de Doutorado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Diego Furtado Silva

Large-scale similarity-based time series mining

Doctoral dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP, in partial fulfillment of the requirements for the degree of the Doctorate Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Gustavo Enrique de Almeida Prado
Alves Batista

Co-advisor: Prof. Dr. Eamonn John Keogh

USP – São Carlos
October 2017

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

SS5861 Silva, Diego Furtado
Large-scale similarity-based time series mining
/ Diego Furtado Silva; orientador Gustavo Enriquede
Almeida Prado Alves Batista; coorientador Eamonn
John Keogh. -- São Carlos, 2017.
185 p.

Tese (Doutorado - Programa de Pós-Graduação em
Ciências de Computação e Matemática Computacional) --
Instituto de Ciências Matemáticas e de Computação,
Universidade de São Paulo, 2017.

1. Time Series. 2. Data Mining. 3. Similarity
Measures. 4. Dynamic Time Warping. I. Batista,
Gustavo Enriquede Almeida Prado Alves , orient. II.
Keogh, Eamonn John , coorient. III. Título.

Diego Furtado Silva

Mineração de séries temporais por similaridade em larga escala

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Gustavo Enrique de Almeida Prado Alves Batista

Coorientador: Prof. Dr. Eamonn John Keogh

USP – São Carlos
Outubro de 2017

ACKNOWLEDGEMENTS

No momento em que escrevo estes agradecimentos, eu me sinto finalizando uma grande etapa, que não começou junto ao início deste trabalho de Doutorado. Esta etapa foi iniciada há muito tempo. Se eu agradecesse a todos que dela participaram, como já fiz em outras ocasiões, essa seria a maior seção desta tese. Entretanto, há alguns agradecimentos que jamais poderiam ser deixados de lado.

Agradeço primeiramente à minha família, que sempre me apoiou em todas as minhas decisões pessoais e profissionais em toda a minha vida. Essa é, sem dúvida, a base de qualquer conquista. Em especial, agradeço aos meus pais, Herminia e Eduardo, que sempre me incentivaram e me deram suporte para o meu desenvolvimento, e à minha esposa, Camila, que esteve ao meu lado em todos os momentos de dificuldades e de alegrias.

Agradeço o meu orientador, Gustavo Batista, com quem eu já trabalho há quase 10 anos e nunca faltou com o meu desenvolvimento acadêmico. Pelo contrário, sempre achou uma brecha no seu corrido cotidiano profissional para atender às minhas dúvidas e ideias e me auxiliar com o andamento do meu trabalho.

Aos colegas do Laboratório de Inteligência Computacional (LABIC), eu devo agradecer por diversos motivos. A famigerada hora do café é, sem dúvida, um grande resumo deles: de bobagens sem sentido a discussões profundas sobre aprendizado de máquina, sem essa interação nada andaria para frente.

I also would like to thank my co-workers and friends from Riverside. Whenever I think about what I learned and experienced in the short period of one year there, I get myself amazed. Especially, thanks to my “co-boss”, Eamonn Keogh, who even being a worldwide recognized researcher is an incredibly accessible and nice person, who immensely collaborated with this work.

Por fim, eu agradeço à CAPES (por meio do processo 1397276) e à FAPESP (processos #2013/26151-5 e #2015/07628-0), por terem contribuído financeiramente com o desenvolvimento deste projeto, e terem me proporcionado a possibilidade de dividir meus resultados com importantes pesquisadores ao redor do mundo.

*“When no idea seems right,
the right one must seem wrong.”
(Marvin Minsk)*

RESUMO

SILVA, D. F. **Mineração de séries temporais por similaridade em larga escala**. 2017. 185 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2017.

Séries temporais são ubíquas no dia-a-dia do ser humano. Dados organizados no tempo são gerados em uma infinidade de domínios de aplicação, como medicina, biologia, economia e processamento de sinais. Devido ao grande interesse nesse tipo de dados, diversos métodos de mineração de dados temporais foram propostos nas últimas décadas. Muitos desses métodos possuem uma característica em comum: em seu núcleo, há uma função de (dis)similaridade utilizada para comparar as séries. *Dynamic Time Warping* (DTW) é indiscutivelmente a medida de distância mais relevante na análise de séries temporais. A principal dificuldade em se utilizar a DTW é seu alto custo computacional. Ao mesmo tempo, algumas tarefas de mineração de séries temporais, como descoberta de *motifs*, requerem um alto número de cálculos de distância. Essas tarefas despendem um grande tempo de execução, mesmo utilizando-se medidas de distância menos custosas, como a distância Euclidiana. Esta tese se concentra no desenvolvimento de algoritmos eficientes que permitem a análise de dados temporais em larga escala, utilizando métodos baseados em similaridade. As contribuições desta tese têm implicações em variadas tarefas de mineração de dados, como classificação, agrupamento e descoberta de padrões frequentes. Especificamente, as principais contribuições desta tese são: (i) um algoritmo para acelerar o cálculo exato da distância DTW e sua incorporação ao processo de busca por similaridade; (ii) um novo algoritmo baseado em DTW para prover invariância a prefixos e sufixos espúrios no cálculo da distância; (iii) uma representação de similaridade musical com implicações em diferentes tarefas de mineração de dados musicais e um algoritmo eficiente para computá-la; (iv) um método eficiente e *anytime* para encontrar *motifs* e *discords* baseado na medida DTW invariante a prefixos e sufixos.

Palavras-chave: Séries Temporais, Mineração de Dados, Medidas de Similaridade, Dynamic Time Warping.

ABSTRACT

SILVA, D. F. **Large-scale similarity-based time series mining**. 2017. 185 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2017.

Time series are ubiquitous in the day-by-day of human beings. A diversity of application domains generate data arranged in time, such as medicine, biology, economics, and signal processing. Due to the great interest in time series, a large variety of methods for mining temporal data has been proposed in recent decades. Several of these methods have one characteristic in common: in their cores, there is a (dis)similarity function used to compare the time series. Dynamic Time Warping (DTW) is arguably the most relevant, studied and applied distance measure for time series analysis. The main drawback of DTW is its computational complexity. At the same time, there are a significant number of data mining tasks, such as motif discovery, which requires a quadratic number of distance computations. These tasks are time intensive even for less expensive distance measures, like the Euclidean Distance. This thesis focus on developing fast algorithms that allow large-scale analysis of temporal data, using similarity-based methods for time series data mining. The contributions of this work have implications in several data mining tasks, such as classification, clustering and motif discovery. Specifically, the main contributions of this thesis are the following: (i) an algorithm to speed up the exact DTW calculation and its embedding into the similarity search procedure; (ii) a novel DTW-based spurious prefix and suffix invariant distance; (iii) a music similarity representation with implications on several music mining tasks, and a fast algorithm to compute it, and; (iv) an efficient and anytime method to find motifs and discords under the proposed prefix and suffix invariant DTW.

Keywords: Time Series, Data Mining, Similarity Measures, Dynamic Time Warping.

LIST OF FIGURES

Figure 1 – The difference between the alignments obtained by the Euclidean distance and Dynamic Time Warping between two time series	28
Figure 2 – Sakoe-Chiba window and Itakura’s parallelogram	29
Figure 3 – Optimal non-linear alignment according to the DTW with and without warping window constraints	30
Figure 4 – Example of dimensionality reduced time series	32
Figure 5 – Examples of possible automatically learned warping windows: <i>RK</i> band and <i>sDTW</i>	33
Figure 6 – The incremental resolution refinement made by the FastDTW algorithm . .	35
Figure 7 – Example of how a prefix may severely interfere in the DTW distance calculation	52
Figure 8 – Three heartbeats taken from a one-minute period of a healthy male	53
Figure 9 – Example of time series collection procedure for the Gun-Point dataset . . .	55
Figure 10 – Example of a time series containing the event to be classified with prefix and suffix	55
Figure 11 – Hierarchical clustering results by using OBE-DTW	58
Figure 12 – Example of a meaningless alignment obtained by OBE-DTW	59
Figure 13 – Clusterings on a toy dataset using the classic DTW, OBE-DTW, and the ψ -DTW	59
Figure 14 – The accuracy after padding the Cricket X dataset with increasing lengths of random walk data	63
Figure 15 – Classification results obtained by ψ -DTW on the Motor Current dataset . .	66
Figure 16 – Classification results obtained by ψ -DTW on the AIBO Robot Surface dataset	66
Figure 17 – Classification results obtained by ψ -DTW on the AIBO Robot Activity dataset	67
Figure 18 – Classification results obtained by ψ -DTW on the Palm Graffiti Digits dataset	67
Figure 19 – Classification results obtained by ψ -DTW on the AUSLAN dataset	68
Figure 20 – Classification results obtained by ψ -DTW on the Human Activity Recognition dataset	69
Figure 21 – The distance between all the pairs of fifty time series objects in the AUSLAN dataset sorted by their DTW distances	70
Figure 22 – Upper and lower sequences of a given query time series estimated by LB_Keogh	71
Figure 23 – Calculation of the LB_Keogh given a time series and its upper and lower envelopes	71

Figure 24 – Calculation of the ψ -LB_Keogh given a time series and its upper and lower envelopes	72
Figure 25 – Optimal non-linear alignment and the matrix obtained by the dynamic programming algorithm	77
Figure 26 – DTW matrix between two time series	81
Figure 27 – Pruning in the lower triangular matrix	83
Figure 28 – Pruning in the upper triangular matrix	84
Figure 29 – Regions of the DTW matrix pruned by our proposed criteria by using the sqED and the true DTW upper bounds for the same pair of time series	86
Figure 30 – Counting of cases in which the best clustering result was obtained by each evaluated warping window length	88
Figure 31 – Time to calculate the all-pairwise DTW distances with different warping window sizes	89
Figure 32 – Accumulative time to calculate the all-pairwise DTW distances with different warping window sizes, when the distance calculated for a smaller window size is used as UB for the next larger window size	91
Figure 33 – DTW between two time series with its cumulative cost matrix and the optimal warping path	97
Figure 34 – LB_{KimFL} and LB_{Keogh} lower bound functions	100
Figure 35 – General scheme of the UCR Suite	101
Figure 36 – DTW matrix between two electrocardiogram subsequences	103
Figure 37 – Strategies adopted by PrunedDTW to prune the beginning and the end of each row of the DTW cumulative cost matrix	104
Figure 38 – Comparison of the Euclidean distance, Dynamic Time Warping and best-so-far values during the similarity search	106
Figure 39 – Example of subsequence in the PAMAP dataset	111
Figure 40 – A fingertip oximeter and five seconds of PPG data obtained by its use	112
Figure 41 – Example of a subsequence describing a freezing of gate episode	113
Figure 42 – Runtime of both UCR and UCR-USP suites on the experimented datasets by varying the query length	114
Figure 43 – Runtime of both UCR and UCR-USP suites on the experimented datasets by varying the warping window length	115
Figure 44 – Speedup ratio in the ECG dataset	116
Figure 45 – Heater monitoring during 24 hours in different seasons	117
Figure 46 – Examples of trajectories of a soccer player monitored during 51.2 seconds, 5 minutes, and 10 minutes	118
Figure 47 – Single-linkage clustering obtained by using DTW with relative warping window lengths of 10% and 50%	120

Figure 48 – Self-distance matrix, the cross-distance matrix between different recordings, and their respective SiMPle	125
Figure 49 – The cross-correlation between a time series B and a subsequence from A . .	127
Figure 50 – Sliding dot-product between a subsequence from the time series A and the time series B reusing previous calculations	128
Figure 51 – Cover song recognition in a streaming fashion by using SiMPle	134
Figure 52 – Runtime obtained by querying one song when varying the number of objects in the dataset and the length of the time series	135
Figure 53 – Average runtime for querying five random queries to the remaining examples on the YouTube Covers dataset by varying the length of the time series, given by the number of features per second	135
Figure 54 – Discord and repeated pattern in the song “Let It Be” by The Beatles	137
Figure 55 – Histogram of the SiMPle index for the song “New York, New York”	137
Figure 56 – Scatter plot of the SiMPle index for the song “Hotel California”	138
Figure 57 – Arc plot for the song “Hotel California”	139
Figure 58 – SiMPle obtained between the songs “Ice Ice Baby” and “Under Pressure” .	140
Figure 59 – Subsequences obtained by the same gestures with warping and endpoints differences	143
Figure 60 – Illustration of a time series with a given subsequence, its nearest neighbor and two examples of trivial matches	144
Figure 61 – Alignment obtained by matching two subsequences by the ED (<i>left</i>) and the DTW (<i>right</i>)	146
Figure 62 – The ψ -DTW allows an elastic matching in which a non-linear alignment is obtained and points in the extremities can be ignored if they do not present a similar pattern	148
Figure 63 – The LB_{ψ} , a lower bound for ψ -DTW	150
Figure 64 – A pair of motifs found by ψ -DTW considered distant by the traditional DTW	152
Figure 65 – A discord found by DTW and its nearest neighbor according the traditional DTW and ψ -DTW	153
Figure 66 – Runtime for calculating the distance matrix profile using a brute force DTW algorithm, our method, and STOMP for different subsequence lengths	154
Figure 67 – Runtime for calculating the distance matrix profile using a brute force DTW algorithm, our method, and STOMP for different time series lengths	155
Figure 68 – First motif pair found by ED and the first pair of ELMO in the athlete positioning data	156
Figure 69 – The discord found by ED and its nearest neighbor in the athlete positioning data	157
Figure 70 – The discord found by ED and its nearest neighbor according the ψ -DTW in the athlete positioning data	157
Figure 71 – ELD with its respective nearest neighbor in the athlete positioning data . . .	157

Figure 72 – Three first motifs according ED and first pairs of ELMO in the motion capture data	158
Figure 73 – Discord discovered by the ED with its nearest neighbors according ED and ψ -DTW in the motion capture data	158
Figure 74 – ELD in the motion capture data and its nearest neighbor	159
Figure 75 – Three first motifs according ED and first pairs of ELMO in the gesture data .	159
Figure 76 – Discord according the ED with its nearest neighbor and the subsequence considered nearest neighbor according the ψ -DTW in the gesture data . . .	160
Figure 77 – ELD in the gesture data and its nearest neighbor	160
Figure 78 – Three first motifs according ED and first pairs of ELMO	161
Figure 79 – Final EMP and the partial values obtained after 5% of the total execution time	162

LIST OF ALGORITHMS

Algoritmo 1 – DTW algorithm	29
Algoritmo 2 – DTW algorithm with warping window	30
Algoritmo 3 – ψ -DTW algorithm	61
Algoritmo 4 – PrunedDTW algorithm	82
Algoritmo 5 – Pruning criteria implementation	83
Algoritmo 6 – SS-PrunedDTW algorithm	107
Algoritmo 7 – Procedure to calculate SiMPle and SiMPle index	127
Algoritmo 8 – SiMPle-Fast	129

LIST OF TABLES

Table 1	– Summary of the accuracy obtained by OBE-DTW, DTW, and ψ -DTW	69
Table 2	– Tightness of LB_Keogh and ψ -LB_Keogh	72
Table 3	– Percentual runtime of FastDTW and PrunedDTW regarding the conventional DTW algorithm	92
Table 4	– Mean average precision, precision at 10, and mean rank of first correctly identified cover on the YoutubeCovers dataset	132
Table 5	– Mean average precision, precision at 10, and mean rank of first correctly identified cover on the Mazurkas dataset	132

CONTENTS

1	INTRODUCTION	25
1.1	Distance Measures for Time Series Comparing	27
1.1.1	<i>Warping Windows</i>	28
1.1.2	<i>Providing Invariances to Distance Measures</i>	31
1.2	Dynamic Time Warping Approximations	31
1.2.1	<i>Dimensionality Reduction</i>	32
1.2.2	<i>Warping Window Learning</i>	33
1.2.3	<i>Lucky Time Warping</i>	34
1.2.4	<i>FastDTW</i>	34
1.2.5	<i>Anytime Clustering by Approximating the Distance Matrix</i>	35
1.2.6	<i>Exploring Time Series Sparsity To Approximate DTW</i>	35
1.3	Time Series Data Mining and Information Retrieval	36
1.3.1	<i>Similarity Search</i>	37
1.3.2	<i>Classification</i>	38
1.3.3	<i>Clustering</i>	40
1.3.4	<i>Motifs and Discord Discovery</i>	41
1.3.5	<i>Information Retrieval</i>	43
1.4	Main Contributions	44
1.5	Thesis Organization	45
1.5.1	<i>Chapter 2</i>	45
1.5.2	<i>Chapter 3</i>	46
1.5.3	<i>Chapter 4</i>	47
1.5.4	<i>Chapter 5</i>	48
1.5.5	<i>Chapter 6</i>	49
1.5.6	<i>Chapter 7</i>	49
1.5.7	<i>Chapter 8</i>	50
2	PREFIX AND SUFFIX INVARIANT DYNAMIC TIME WARPING .	51
2.1	Introduction	51
2.2	Time Series Suffix and Prefix	54
2.3	Definitions and Background	56
2.4	Related Work	57
2.5	Prefix and Suffix-Invariant DTW (ψ -DTW)	59

2.6	Experimental Evaluation	62
2.6.1	<i>The Effect of ψ-DTW on Different Lengths of Endpoints</i>	63
2.6.2	<i>Case Studies</i>	64
2.6.2.1	<i>Motor Current Data</i>	65
2.6.2.2	<i>Robot Surface and Activity Recognition</i>	65
2.6.2.3	<i>Gesture Recognition</i>	67
2.6.2.4	<i>Sign Language Recognition</i>	68
2.6.2.5	<i>Human Activity Recognition</i>	68
2.6.2.6	<i>Summary of the Results</i>	68
2.7	Lower Bounding ψ -DTW	69
2.8	Conclusion	73
3	SPEEDING UP ALL-PAIRWISE DYNAMIC TIME WARPING MA- TRIX CALCULATION	75
3.1	Introduction	75
3.2	Background	76
3.3	On the Need of the All-Pairwise Distance Matrix	78
3.4	Related Work	78
3.4.1	<i>Similarity Search</i>	78
3.4.2	<i>DTW Approximations</i>	79
3.4.3	<i>Biological Sequences Alignment</i>	79
3.4.4	<i>FTW</i>	80
3.5	DTW with Pruned Warping Paths	80
3.5.1	<i>The Intuition Behind our Proposal</i>	81
3.5.2	<i>Pruning Strategy</i>	81
3.5.3	<i>Iteratively Updating the Upper Bound</i>	84
3.5.4	<i>Other UB Approaches</i>	85
3.6	Experimental Evaluation	86
3.6.1	<i>Benchmark Datasets</i>	87
3.6.2	<i>On the Warping Window Length</i>	87
3.6.3	<i>Runtime for All-Pairwise DTW Matrices</i>	88
3.6.4	<i>Accumulative Runtime for All-Pairwise DTW Matrices</i>	90
3.6.5	<i>Comparison with FastDTW</i>	90
3.7	Conclusion and Future Work	92
4	SPEEDING UP SIMILARITY SEARCH UNDER DTW BY PRUN- ING UNPROMISING ALIGNMENTS	93
4.1	Introduction	93
4.2	Background and Definitions	95

4.3	The UCR Suite	98
4.4	DTW with Pruned Warping Paths	102
4.4.1	<i>The Intuition Behind PrunedDTW and its Pruning Strategies</i> . . .	102
4.4.2	<i>Embbding PrunedDTW into the Similarity Search Procedure</i> . . .	104
4.4.3	<i>On the Correctness of the SS-PrunedDTW</i>	109
4.5	Experimental Evaluation	110
4.5.1	<i>Data</i>	111
4.5.1.1	<i>Physical Activity Monitoring</i>	111
4.5.1.2	<i>Athletic Performance Monitoring</i>	111
4.5.1.3	<i>Electrocardiography</i>	111
4.5.1.4	<i>Photoplethysmography</i>	112
4.5.1.5	<i>Freezing of Gait</i>	112
4.5.1.6	<i>Electrical Load Measurements</i>	112
4.5.2	<i>Results and Discussion</i>	113
4.6	On the Need of Long Queries and Large Warping Windows	116
4.6.1	<i>Query Length</i>	117
4.6.2	<i>Warping Window</i>	118
4.7	Pruning Paths on DTW Variations and Other Distance Measures .	120
4.8	Conclusion	121
5	FAST SIMILARITY MATRIX PROFILE FOR MUSIC ANALYSIS AND EXPLORATION	123
5.1	Introduction	123
5.2	SiMPle: Similarity Matrix Profile	125
5.3	SiMPle-Based Cover Song Recognition	128
5.3.1	<i>On the Structural Invariance</i>	130
5.3.2	<i>Experimental Evaluation</i>	130
5.3.2.1	<i>Datasets</i>	131
5.3.2.2	<i>Results and Discussion</i>	131
5.3.2.3	<i>Streaming Cover Song Recognition</i>	133
5.4	Scalability	134
5.5	Music Data Exploration using SiMPle	136
5.5.1	<i>Discord and Repeated Patterns</i>	136
5.5.2	<i>Audio Thumbnailing</i>	137
5.5.3	<i>Visualization</i>	138
5.5.4	<i>Endless Reproduction</i>	139
5.5.5	<i>Sampling Identification</i>	139
5.6	Conclusion	140
6	ELASTIC TIME SERIES MOTIFS AND DISCORDS	141

6.1	Introduction	141
6.2	Definitions and Background	143
6.3	Proposed Method	148
6.3.1	<i>Online Normalization</i>	149
6.3.2	<i>Lower Bounding</i>	149
6.3.3	<i>Early Abandoning</i>	150
6.3.4	<i>Exploring the Symmetry of ψ-DTW</i>	151
6.3.5	<i>Heuristic Order</i>	151
6.4	Why ψ -DTW Instead of the Regular DTW?	152
6.5	Experimental Evaluation	153
6.5.1	<i>Scalability</i>	153
6.5.2	<i>Case studies</i>	155
6.5.2.1	<i>Athlete Positioning</i>	156
6.5.2.2	<i>Motion Capture</i>	157
6.5.2.3	<i>Gesture Analysis</i>	159
6.5.2.4	<i>Music Processing</i>	160
6.5.3	<i>Anytime ELMO discovery</i>	161
6.6	Conclusion	162
7	OTHER CONTRIBUTIONS	165
8	CONCLUSION	169
	BIBLIOGRAPHY	173

INTRODUCTION

Time series are ubiquitous in the day-by-day of human beings. A diversity of application domains such as medicine, biology, economics, and signal processing generate data arranged in time. Consequently, the time series analysis has attracted the attention and effort of many researchers around the world. Due to the great interest in time series, many methods for analyzing this kind of data have been proposed for different temporal data mining tasks, such as classification, clustering, motif discovery, and anomaly detection. Several of these methods have one attribute in common: in their cores, there is a (dis)similarity function used as the principal means to compare the time series.

Dynamic Time Warping (DTW) is arguably the most relevant distance measure for time series analysis. Such relevance has been evidenced, for instance, by a large body of experimental research showing that similarity search with DTW is a very effective classification procedure. Specifically, the 1-nearest neighbor DTW (1-NN-DTW) algorithm frequently outperforms more sophisticated methods on a large set of benchmark datasets ([DING *et al.*, 2008](#); [WANG *et al.*, 2013](#)). This relevance is also demonstrated in other tasks, such as clustering ([ULANOVA; BEGUM; KEOGH, 2015](#)), and is pointed as future direction in some others, like in the motif discovery ([TORKAMANI; LOHWEG, 2017](#)).

DTW allows a nonlinear matching between time series observations, known as *warping*. In summary, while a non-elastic distance such as Euclidean distance requires that two observations a and b must occur at the same time $t_a = t_b$ to be matched, DTW allows the matching between observations at different times. In other words, $t_a \cong t_b$, such that the sum of the differences between matched observations is minimal.

The main drawback of DTW is its computational complexity. The algorithm to calculate DTW is a dynamic programming technique that requires a quadratic matrix concerning the length of the time series. Although we can use a simple trick to calculate DTW in linear space complexity, there is no known exact algorithm to reduce its time complexity.

DTW does not obey the triangular inequality. Therefore, it cannot be indexed by a multitude of metric access methods (MAM) proposed in the literature (CHÁVEZ *et al.*, 2001). In contrast, the community has offered a handful of lower-bounding distances that allow DTW to be indexed using the GEMINI framework (FALOUTSOS; RANGANATHAN; MANOLOPOULOS, 1994). Certainly, LB_Keogh is the most well-known lower-bounding distance for DTW and has demonstrated to be very effective to speed up similarity search (KEOGH; RATANAMAHATANA, 2005; KEOGH *et al.*, 2006; RAKTHANMANON *et al.*, 2012).

It is important to notice that the majority of the techniques proposed to accelerate time series mining algorithms are restricted to the task of similarity search. In this case, delimiting the space of possible neighbors of a query requires most of the computational effort. To the best of our knowledge, there is no method capable of speeding up the exact DTW calculation without using some statistics calculated for indexing purposes. However, there is a significant number of data mining algorithms, including clustering, motif discovery, and classification that requires the distance between plenty of pairs of time series, or even the all-pairwise distance matrix.

In these cases, the use of DTW becomes impracticable. If a researcher or practitioner is interested in applying DTW with algorithms that require the all-pairwise distance matrix, the only speed up techniques available are warping windows (also known as constraint bands) or DTW approximations.

Warping windows (ITAKURA, 1975; SAKOE; CHIBA, 1978) are very useful in practice since they limit the difference in time between two matched observations, avoiding unrepresentative matchings. However, the exact window size that would provide the best results for a dataset at hand is data dependent. For classification by the nearest neighbor algorithm, some authors have provided empirical evidence that such optimal band sizes are usually small (less than 10% of the time series length) (RATANAMAHATANA; KEOGH, 2005). For tasks outside classification there are no clear guidelines, except some recent proposals such as the use of label information in a semi-supervised procedure to improving clustering under DTW (DAU; BEGUM; KEOGH, 2016). Otherwise, the best one can do is to execute the algorithm with different window sizes and interpret the results.

Approximate DTW is an approach that trades quality for speed. DTW approximations are usually sub-quadratic or even linear in time but provide no guarantees regarding quality. Most of them calculate an approximate distance measure between a pair of time series (SALVADOR; CHAN, 2007; SPIEGEL; JAIN; ALBAYRAK, 2014), but some work has cast the all pairwise DTW matrix calculation to the anytime framework (ZHU *et al.*, 2012).

This thesis summarizes the research results of similarity-based methods for time series data mining. Specifically, this research focused on developing scalable and accurate algorithms based on the distance between pairs of time series. The main contributions of this work are a fast algorithm to calculate the exact DTW and its embedding in the similarity search task, a novel spurious endpoints-invariant distance for time series, and new advances in motifs and discord

discovery. Before introducing these contributions, we present the necessary background and related work.

The remaining of this chapter is organized as follows. Initially, Section 1.1 introduces basic concepts on distance measures to compare time series data, including the Euclidean and the DTW distances, as well as some improvements on these distance measures. Section 1.2 presents some methods to approximate the DTW distance, a standard approach to obtain a fast response for time series mining algorithms. Some of the most important tasks and algorithms for temporal data mining are discussed in Section 1.3. Section 1.4 briefly describes the main contributions of this thesis concerning the topics discussed in the previous sections. Finally, Section 1.5 concludes this chapter by revealing the organization of this thesis.

1.1 Distance Measures for Time Series Comparing

The most established family of distance measures for dissimilarity comparison between time series are the metrics based on the Minkowski distance. Equation 1.1 defines the Minkowski distance for two time series x and y .

$$d_{minkowski}(x, y) = \left[\sum_{k=1}^N (x_k - y_k)^p \right]^{1/p} \quad (1.1)$$

where N is the length of both x and y time series, i.e., the number of data points (also known as observations). The values x_k and y_k are the observations of those time series in the k -th position, and $p \in \mathbb{N}$ is a parameter defined by the user.

Specifically, the Euclidean distance (ED) is the most applied distance derived from the Minkowski distance. ED is obtained by setting $p = 2$ in Equation 1.1. Formally, Equation 1.2 defines the ED between time series x and y .

$$ed(x, y) = \sqrt{\sum_{k=1}^N (x_k - y_k)^2} \quad (1.2)$$

Therefore, ED – as well as any measure derived from Minkowski distance – is only applicable when the time series objects under comparison have the same number of observations. For an extensive review of distance measures for time series, we refer the reader to (GIUSTI; BATISTA, 2013).

The ED measures the dissimilarity between time series comparing the observations at the same time. For this reason, it can be susceptible to distortions in the time axis (KEOGH; RATANAMAHATANA, 2005). Many applications require a more flexible matching of observa-

tions. The DTW distance achieves an optimal nonlinear alignment of the observations under the following constraints (RATANAMAHATANA; KEOGH, 2004)¹:

- **Boundary constraint.** The matching occurs for the whole time series x and y . Therefore it starts at $(1, 1)$ and ends at (N, M) ;
- **Monotonicity constraint.** The relative order of the observations has to be preserved;
- **Continuity constraint.** The matching is made in one-unit steps. It means that the matching never skips one or more observations.

Figure 1 exemplifies the difference between the linear alignment obtained by ED and the nonlinear alignment achieved by DTW.

Figure 1 – The difference between the alignments obtained by the Euclidean distance (*left*) and DTW (*right*) between two time series. Note that the time series present a significantly different offset. However, it was made only for visualization purposes. In general, this difference needs to be treated in order to calculate an appropriate distance measure (c.f. Section 1.1.2)

Source: Elaborated by the author.

DTW is usually calculated using a dynamic programming algorithm. To evaluate the match between each pair of observations, this algorithm iteratively minimizes each subproblem by using a quadratic cumulative cost matrix. A subproblem, in this case, is the partial match of a subsequence from each object under comparison, starting from the first observation. For clarity, Algorithm 1 presents the procedure to find the DTW distance between two time series.

1.1.1 Warping Windows

Warping window, or constraint band, is a simple, well-known and widely used approach to speed up DTW. Warping windows define the maximum allowed time difference between two matched observations. From the algorithm standpoint, this technique restricts the cells that need to be computed to a smaller area around the main diagonal of the matrix.

Figure 2 shows examples of two well-known warping window: Sakoe-Chiba window (SAKOE; CHIBA, 1978) and Itakura's parallelogram (ITAKURA, 1975).

¹ For the remaining of this section, we assume that the time series objects may have different lengths. Therefore, $x = x_1, x_2, \dots, x_N$ and $y = y_1, y_2, \dots, y_M$.

Algorithm 1 – DTW algorithm**Requer:** Time series x , with length N ; Time series y , with length M **Assegure:** The distance between x and y according DTW

▷ Initialize the matrix of DTW calculations

```

1: para  $i \leftarrow 1$  até  $N$  faça
2:    $D[i, 0] \leftarrow \infty$ 
3: fim para
4: para  $i \leftarrow 1$  até  $M$  faça
5:    $D[0, i] \leftarrow \infty$ 
6: fim para
7:  $D[0, 0] \leftarrow 0$ 
8: para  $i \leftarrow 1$  até  $N$  faça
9:   para  $j \leftarrow 1$  até  $N$  faça
10:     $D[i, j] = sqED(x_i, y_j) + \min(D[i-1, j-1], D[i-1, j], D[i, j-1])$ 
11:   fim para
12: fim para
    retorna  $D[N, M]$ 

```

Figure 2 – Sakoe-Chiba window (*left*) and Itakura's parallelogram (*right*). The area marked in gray represents all the cells of the cumulative cost matrix that should be computed. The position $(0,0)$ of the matrix is represented in the top-left position and the last column and row is located in the bottom-right position

Source: Elaborated by the author.

For clarity, Algorithm 2 presents an implementation of DTW with the Sakoe-Chiba warping window. The only difference from this algorithm to Algorithm 1 is the definition of the range of values constrained by the warping window, inducing subtle modifications from line 9 to the definition of the second loop (line 11).

Warping windows improve efficiency since there is no need to calculate all the cells of the DTW matrix. In some cases, the use of warping windows can also help to improve the quality of the matching between the time series. These windows may help to avoid spurious matches, i.e., to avoid that DTW forcefully considers two very different objects as similar.

For a clearer understanding, consider two series that are mostly constant, except for the appearance of a single peak in their trajectories. The unrestricted DTW will likely return a low distance value, indicating that the series are similar. On the other hand, the DTW with warping

Algorithm 2 – DTW algorithm with warping window**Requer:** Time series x , with length N ; Time series y , with length M ; Warping window length w **Assure:** The distance between x and y according DTW

▷ Initialize the matrix of DTW calculations

```

1: para  $i \leftarrow 1$  até  $N$  faça
2:    $D[i, 0] \leftarrow \infty$ 
3: fim para
4: para  $i \leftarrow 1$  até  $M$  faça
5:    $D[0, i] \leftarrow \infty$ 
6: fim para
7:  $D[0, 0] \leftarrow 0$ 
8: para  $i \leftarrow 1$  até  $N$  faça
9:    $begWin \leftarrow \max(1, i - w)$ 
10:   $endWin \leftarrow \min(M, i + w)$ 
11:  para  $j \leftarrow begWin$  até  $endWin$  faça
12:     $D[i, j] = sqED(x_i, y_j) + \min(D[i - 1, j - 1], D[i - 1, j], D[i, j - 1])$ 
13:  fim para
14: fim para
    retorna  $D[N, M]$ 

```

window would consider the pair of time series distant. Figure 3-left illustrates an extreme case, in which the peaks occur at very different times. In contrast, Figure 3-right shows the matching for the same time series with DTW restricted by constraint bands. In the second case, the distance is greater than in the first one.

Figure 3 – Optimal non-linear alignment according to the DTW without (*left*) and with (*right*) warping window constraints

Source: Elaborated by the author.

Note that, in this example, we cannot say *a priori* if those series are different (representing different phenomena or classes) or similar but out of phase. Therefore, it is clear that the size of the warping windows is data dependent. As stated earlier in this chapter, finding the best warping window is an open problem on time series mining.

1.1.2 Providing Invariances to Distance Measures

While a large body of evidence points to the fact that DTW is suitable for many scenarios, its performance may depend on the application domain. Specifically, one should look at the invariances required by the domain it will be applied.

For instance, consider a pair of time series from the same hand gesture. If one of these movements starts from a different point in the space, any distance measure is likely to consider these gestures completely different each other. The same can be said if one of the gestures is “wider” than the other one. In these cases, we are required to deal with offset and amplitude, respectively. To provide invariance to both effects, we can simply z-normalize each of the time series (FALOUTSOS; RANGANATHAN; MANOLOPOULOS, 1994). This procedure transforms the time series such that the observations in the new data have mean $\mu = 0$ and standard deviation $\sigma = 1$.

Considering the same example, providing invariance to amplitude and offset may be not sufficient. Gesture data commonly suffers from local scaling, given by different paces when performing the gestures. These distortions are also called warping and, as we have previously noticed, the DTW naturally provides invariance to this effect (DING *et al.*, 2008).

In addition to the mentioned distortions, time series may suffer from uniform scaling (KEOGH, 2003), occlusion (VLACHOS *et al.*, 2003), and differences in terms of phase (ZUNIC; ROSIN; KOPANJA, 2006; KEOGH *et al.*, 2009) and complexity (BATISTA *et al.*, 2014). Usually, similarity-based time series mining algorithms need to deal with multiple invariances. In other words, depending on the application domain, two or more of these invariances may be required. For a complete review on the pointed invariances, we refer the reader to (BATISTA *et al.*, 2014).

1.2 Dynamic Time Warping Approximations

Distance measures that approximate DTW are a popular approach to speed up DTW calculations. Most of these approximations are quadratic with small constants or sub-quadratic in time complexity. Since these methods usually only require two time series objects as input, they are directly applicable to any data mining task, including cases that require calculating the all-pairwise distance matrix.

The main drawback of this approach is the fact that it may compromise the quality of the mining algorithm in favor of time. Most approaches do not provide any guarantees regarding quality. In other words, the user has no means of setting a maximum allowable error with respect to the true DTW.

In this section, we describe some approaches to approximate the DTW distance.

1.2.1 Dimensionality Reduction

A straightforward approach to speed up DTW is to calculate the distance on a time series with reduced dimensionality. For example, a time series can be segmented into equal-sized frames, in which a single value replaces all the observations in the frame using the average of the frame observations. This procedure creates a new time series with a lower number of observations (KEOGH; PAZZANI, 2000). Figure 4 exemplifies this method, called Piecewise Approximation Aggregation (PAA).

Figure 4 – Example of dimensionality reduced time series. The original (blue) time series has 50 observation, while the reduced (red) has only 10, define by the upper axis

Source: Elaborated by the author.

Let n be the desired length of the reduced time series. For simplicity, a factor of N , the length of the original time series x , is used to defined a value for n . According to PAA, a time series $\bar{x}$ is composed by the n observations defined of Equation 1.3.

The PAA only requires the parameter n as input, which is the length of the resulting reduced time series. Equation 1.3 defines the reducing procedure of a time series s , considering n as a factor of N .

$$\bar{x}_i = \frac{n}{N} \sum_{j=\frac{N}{n}(i-1)+1}^{\frac{N}{n}i} x_j \quad (1.3)$$

Note that the gain in efficiency obtained by PAA is directly proportional to the length of the frame used to reduce the original data. If the frames are small, then the time series length is not significantly reduced. Therefore, the achieved speedup is small. If the frame size is large, we can expect to have a much higher speedup, at the cost of a higher error.

In addition to PAA, the literature has several approaches to create reduced time series representations. Some examples are Discrete Fourier Transform (DFT) (AGRAWAL; FALOUTSOS; SWAMI, 1993), Singular Value Decomposition (SVD) (KORN; JAGADISH; FALOUTSOS, 1997), Discrete Wavelet Transform (DWT) (CHAN; FU, 1999), Adaptive Piecewise Constant Approximation (APCA) (KEOGH *et al.*, 2001), and Symbolic Aggregate approXimation (SAX) (LIN *et al.*, 2003). For a complete description of these time series representation, we refer the reader to (MITSA, 2010).

1.2.2 Warping Window Learning

As previously stated, warping windows are widely applied to accelerate the computation of the DTW distance. For this reason, defining a region of the DTW cost matrix where the optimal warping path is most likely to belong may accelerate the distance calculation. This strategy provides a value that approximates the actual DTW and may improve the mining results depending on the choice of the window length. However, the optimal value for this parameter is difficult to estimate. Also, the Sakoe-Chiba band (SAKOE; CHIBA, 1978) is the most common choice in practice, due to its simplicity, but there are no stated guarantees that this is the optimal choice.

With this in mind, some researchers have proposed the automatic learning of warping windows with arbitrary shapes. The first work that proposed these warping constraints, called *RK* bands (RATANAMAHATANA; KEOGH, 2004), uses different constraints for each class. Such constraints are estimated according to the characteristics observed in the training examples of each class. This fact restricts the use of *RK* bands to classification problems.

More recently, Candan *et al.* (2012) proposed the use of heuristics as the main step of an algorithm to estimate different warping constraints for each pair of time series. These heuristics are based on the distance between points of interest of the time series, estimated by the Scale-Invariant Feature Transform (SIFT) algorithm (LOWE, 1999). This algorithm, called *sDTW*, brings an important advantage to the *RK* bands: it is not necessary to preprocess the whole set of objects to estimate the warping window setting. The algorithm *sDTW* estimates the warping window using only information of the pair of time series at hands. Besides, this approach does not necessarily build a path around the main diagonal of the DTW matrix.

To elucidate these strategies, Figure 5 shows examples of possible warping windows obtained by each approach.

Figure 5 – Examples of possible automatically learned warping windows: *RK* band (left) and *sDTW* (right). The areas highlighted in gray represent all the cells to be calculated in the DTW matrix

Source: Elaborated by the author.

1.2.3 Lucky Time Warping

The Lucky Time Warping (LTW) measure (SPIEGEL; JAIN; ALBAYRAK, 2014) uses a greedy algorithm to estimate a DTW approximation. Specifically, LTW uses a best-first approach to find a sub-optimal alignment between the observations of two time series.

The algorithm starts at the cell $(0,0)$ of the DTW matrix. At each iteration of the algorithm, LTW evaluates each of the three possible next movements for the warping path. Specifically, from a pair of observations (i, j) it assess the pairs $(i+1, j+1)$, $(i+1, j)$, and $(i, j+1)$. The algorithm expands its alignment path to that cell that presents the smaller cost, given by the Euclidean distance between the candidate match. In other words, the algorithm expands the warping path according to the minimal value between $ed(x_i, y_{j+1})$, $ed(x_{i+1}, y_j)$, and $ed(x_{i+1}, y_{j+1})$.

In the worst case, LTW expands exact $N + M$ cells, being N and M the length of the time series under comparison. Therefore, LTW has the convenience of being a linear time approximation of DTW. Even more, LTW is compatible with constraint bands, simply by limiting the space of decision to expand the algorithm in each iteration, i.e., if the expansion extrapolates the region defined by the warping window, its cost is set to infinite. In contrast, the approach lacks any warranties that the greedy path is a reasonable estimate of the optimal path.

1.2.4 FastDTW

FastDTW is probably the best-known approximation of DTW (SALVADOR; CHAN, 2007). This approach uses increasing levels of granularity to represent the time series. In other words, the approach starts in a coarse representation of the time series and iteratively improves the representation until it reaches the original time series. The algorithm uses a dimensionality reduction technique similar to PAA (KEOGH; PAZZANI, 2000) to find these representations.

FastDTW computes a constrained DTW algorithm at each level of granularity. Figure 6 illustrate this process. In the first step, the DTW is calculated over the lowest resolution. The optimal warping path found in this step is used to impose a warping constraint over the next iteration. In the second step, the DTW algorithm runs on the projected warping path, represented by dark gray cells. The algorithm continues until the resolution matches the original time series.

FastDTW is an approximate algorithm because the optimal warping path may be located outside the warping constraint in each step. In an attempt to increase the chances of having the optimal path in the constrained area, FastDTW has a parameter r that expands the warping constraint by an additional number of cells. These cells are colored in light gray in Figure 6.

A time complexity analysis of FastDTW shows that it performs linearly on the length of the time series. However, similarly to other approaches, it has no bounds for the approximation error to the true DTW. An additional drawback is that FastDTW is only an approximation to the unconstrained DTW. In other words, the original algorithm has no support for warping windows.

Figure 6 – The incremental resolution refinement made by the FastDTW algorithm. The dark shaded area represents constraint imposed by the previous level of granularity. The light gray cells are the additional cells that will be analyzed due to the parameter r

Source – (SALVADOR; CHAN, 2007)

1.2.5 Anytime Clustering by Approximating the Distance Matrix

Differently from the previous approaches, (ZHU *et al.*, 2012) proposes an anytime clustering framework for time series using the DTW distance. The idea is to provide an estimate of the all-pairwise distance matrix in an anytime fashion. In a high level of definition, an algorithm is said to be anytime if it may be interrupted at any step (except by a short setup time) and provide the best answer found so far (ZILBERSTEIN, 1996). Ideally, as more time the algorithm has to provide an output, better the quality of the answer.

The algorithm proposed by (ZHU *et al.*, 2012) first initializes the distance matrix with estimated values for DTW. The authors use the idea that ED and LB_Keogh are upper and lower bounds for DTW, respectively, to calculate the estimates. In this way, the first approximation is given by the value of LB_Keogh summed to a ratio between this lower bound and the ED.

From this moment, the algorithm can be interrupted at any time during the execution to provide an estimate of the all-pairwise distance matrix. In each iteration, the distance calculation algorithm chooses one matrix cell storing an estimated distance and replaces it by the true DTW distance. So, in each iteration, the approximate all-pairwise distance matrix is set to be closer to the actual distance matrix.

1.2.6 Exploring Time Series Sparsity To Approximate DTW

A recent strategy to approximate DTW takes advantage of sparsity in time series data (MUEEN *et al.*, 2016). Several novel applications capture time series with few events occurring in a long period. It creates time series composed of long sequences of zero-valued observations and few points with positive values. One example of this phenomenon is the monitoring of social media activity. Consider a user of the social network Twitter. The time series of his/her activity is given by 0 or 1 in each observation, representing no posting event or a tweet made at that moment, respectively. This user probably posts a very few times per day in the micro-blogging platform. However, Twitter records the activity of each user in the order of milliseconds, resulting in very sparse time series.

The main issue of using DTW on sparse data is the high number of comparisons between observations that do not add any cost to the distance, i.e., unnecessary calculations. Another issue is the memory consumption to store so many repeated values. We can easily cope with the later issue by encoding the time series accordingly. For instance, the time series $X = \{4, 0, 0, 0, 2, 0, 0, 5, 0, 0, 0, 0, 3\}$ can be similarly represented by $X = \{4, (3), 2, (2), 5, (4), 3\}$, where the numbers in parenthesis represent the length of the sequence of zero-valued observations. In this simple example, a time series with 13 observations is represented by 7 values.

The intuition behind taking advantage of sparsity to speed up the DTW is to compute the comparisons between zero-valued observations in “large blocks.” For this purpose, [Mueen et al. \(2016\)](#) proposed the algorithm AWarp, which extends DTW to deal with the mentioned representation. The only difference to the original DTW is the way to calculate the cost of matching. In this case, the cost depends on the relation between the compared values. In other words, the cost of matching is calculated in different ways for the cases where both compared values are non-zero values, one of them is a block of zero-valued observations, and both compared values are blocks of zeros.

In addition to propose AWarp, [Mueen et al. \(2016\)](#) present its extension to multidimensional case and warping-constrained DTW. Moreover, the authors prove that the AWarp is exact for binary time series, such as the previously mentioned Twitter’s activity data. When the time series is composed of arbitrary values, AWarp obtains an approximate value.

An experimental evaluation of the method demonstrates the intuitive gain in runtime proportional to the sparsity ratio, i.e., the proportion of zero-valued observations in the time series. In other words, the sparser is the time series, the faster is AWarp. When the number of zeros is three times higher than the number of non-zeros, the AWarp runs two times faster than the traditional DTW. When the number of zero-valued observations is 746 times greater than the remaining values, AWarp achieves a speedup ratio of 557.

Because of its impressive speedup on sparse time series, AWarp was successfully applied to compare time series from Twitter activity to identify bots ([CHAVOSHI; HAMOONI; MUEEN, 2016](#)). However, the experimental results obtained by [Mueen et al. \(2016\)](#) show that AWarp is only suitable for long and sparse time series. Besides, AWarp for non-binary time series is an approximate algorithm which the approximation error is inversely proportional to the sparsity factor.

1.3 Time Series Data Mining and Information Retrieval

Many algorithms have been proposed and studied to analyze temporal data. This section presents some of the most important and used time series mining tasks, as well as relevant work related to each of these tasks. There is an infinity of data mining tasks applied on the time series domain. For this reason, we limited this section to the description of the mining tasks referred in

the next chapters of this thesis. In addition, we discuss how time series are related to the field of information retrieval.

1.3.1 Similarity Search

The similarity search is arguably the most used time series mining task. The goal of the similarity search is to find the most similar objects to a query object in a dataset.

Regarding temporal data, the similarity search requires a (possibly short) time series as query and searches for its nearest neighbors in the reference dataset. The reference dataset may be: (i) a set of short/segmented time series which, in general, are equal-length; (ii) a long time series, usually obtained by a streaming fashion collection procedure. In the first case, the similarity search compares the query to each of the objects in the reference dataset, one by one. In the second one, the search procedure needs to swipe the long time series using a sliding window and comparing the query to the subsequences defined by the observations in each window.

The majority of the papers on this task propose approaches to find the k most similar objects to a query time series with a reduced runtime. These methods usually prune or early abandon a significant number of distance calculations.

Using distance measures based on a linear alignment in the similarity search allows us to use of several well-known metric access methods (CHÁVEZ *et al.*, 2001) and other techniques, such as fast algorithms to calculate cross-correlation (MUEEN; HAMOONI; ESTRADA, 2014), to accelerate the search procedure. However, none of these techniques are applicable to DTW. In this case, the usual procedure is to use lower bound functions that allow DTW to be indexed using the GEMINI framework (FALOUTSOS; RANGANATHAN; MANOLOPOULOS, 1994).

The central idea is to find a lower bound distance $\widehat{dtw}$ such that $\widehat{dtw}(x, y) \leq dtw(x, y)$. In order to be useful, $\widehat{dtw}$ must be computed in less than $O(NM)$ steps, typically linearly.

For simplicity, consider the case we want to find the nearest neighbor of a query object q . Suppose that a variable *best-so-far* stores the true DTW distance to the closest object known up to a certain moment of the search. Evidently, $best-so-far = \infty$ is a proper initialization. Whenever a object (or subsequence) o_k from the reference data is considered as candidate to nearest neighbor, $\widehat{dtw}(q, o_k)$ is computed. Since $\widehat{dtw}$ is a lower bound to the true DTW distance, if $\widehat{dtw}(q, o_k) > best-so-far$ then o_k is certainly more distant than the current closest object. Therefore, the object o_k can be pruned without the expensive calculation of $dtw(q, o_k)$. If $\widehat{dtw}(q, o_k) \leq best-so-far$ then nothing can be concluded and $dtw(q, o_k)$ must be computed to verify if o_k will become the nearest neighbor of q . If o_k is confirmed as the nearest neighbor, then *best-so-far* must be adjusted accordingly. This scheme can be trivially generalized to k -nearest neighbors by setting *best-so-far* as the distance to the k -th neighbor.

Another frequently used technique is to early stop the calculation of the actual or the lower bound DTW measures as soon as their partial sums become greater than *best-so-far*.

Although the general idea is straightforward, some papers use them as intermediate steps in more elaborate schemes ([ASSENT *et al.*, 2009](#); [RAKTHANMANON *et al.*, 2012](#)).

Based on these ideas, many algorithms were proposed to make a fast search of time series using the DTW distance. For a complete description of techniques to speed up the similarity search under DTW, we refer the reader to the Chapter 4.

While the search for similar subsequences may be used standalone for the exploratory data analysis, algorithms often use this procedure as an intermediate step to solve a certain mining task. This is the case, for instance, of classification, clustering, and motifs and discords discovery, the data mining tasks presented in the next sections.

1.3.2 Classification

Classification is the automatic categorization of a single or a group of objects (or examples). In a practical standpoint, the objective of this task is to create a classification model capable of assigning a class labels for new (previously unknown) objects.

A classification model is created from a set of previously labeled examples, called training set. The label of an object may be defined by a domain expert or can be directly obtained by the data collection setup. When the label of a new object is required, the classification model needs to decide what is the most appropriate label based on the classes of the previously known examples, i.e., the training set.

For instance, consider a problem in which we are interested in classify everyday activities using accelerometers ([LONG; YIN; AARTS, 2009](#)). This task may be interesting, for example, to automatically adjust mobile phone configurations or to choose a music genre to play according to the current activity. In this case, the class labels may be “walking,” “driving,” “running,” among others. The training set may be collected by pre-defined sections of each activity of interest, labeling the data accordingly.

The k -nearest neighbor (k -NN) is one of the simplest classification methods. This algorithm considers that the class label of a new object is the same as the label of their k most similar examples in the training set, according to a predefined (dis)similarity measure. If these neighbors have different classes, the mode of the neighbors’ labels is used to classify the object.

Surprisingly, the k -NN algorithm has been successfully used for decades to perform time series classification. Moreover, there exists solid evidence that the results obtained by the simple 1-nearest neighbor with DTW are hard to overcome, even by more elaborated classical methods, such as Random Forests and Support Vectors Machine ([BAGNALL *et al.*, 2017](#)). A study on the performance of distance measures for 1-NN-based classification of time series ([LINES; BAGNALL, 2015](#)) demonstrated that no other known “elastic” distance measure significantly outperforms DTW. The authors demonstrate that is possible to achieve better results using an

ensemble of distance measures – including DTW –, what trades a better accuracy by the time cost of the classification procedure.

Another popular approach to classify time series is based on creating an attribute-value table (LIN; KHADE; LI, 2012; LINES *et al.*, 2012; KATE, 2016; SCHÄFER; LESER, 2017). This table is the most common way to represent data for mining applications. Each row of this table accounts for a single training example and their columns bring the values of the attributes (also known as features, properties, or characteristics) to describe it. This representation can be used as input for a multitude of algorithms to learn a classification model.

An example of this procedure is the method proposed by Kate (2016). The basic idea is to use the DTW distance to the training examples as attribute values. For clarity, consider a training set containing n time series objects. For each object o_i , the algorithm constructs a set of n numerical attributes, whose values are the distance from the object o_i to each of the n training examples. At the end of this procedure, we have at hands a $n \times n$ attribute-value table, that can be used together with several learning methods. When the class label of a new object is required, the classifier calculates its distance to each training time series. These values are used to compose a n -dimensional attribute-value vector, that is submitted to the previously created classification model.

In the same work, Kate (2016) extended the idea to use more than a single distance measure as features. This approach creates a $n \times dn$ attribute-value table, in which d is the number of different distance measures. For this, the distance from each object to all the training examples is measured d times, one for each different measure. The experimental evaluation showed that some combinations of distance measures, such as DTW with and without warping windows, can achieve a statistically significant improvement over the 1-NN with DTW when a Support Vector Machine (STEINWART; CHRISTMANN, 2008) learns the classification model.

In fact, Schäfer and Leser (2017) show that some methods based on transforming time series in an attribute-value table may overcome the 1-NN DTW. Specifically, the authors report statistical difference obtained by BOSS (SCHÄFER, 2015), shapelet transform (LINES *et al.*, 2012), and WEASEL (SCHÄFER; LESER, 2017).

In summary, recent advances in time series classification show that it is an open problem. While novel methods suppress the widely used 1-NN, recent work claims that better methods to learn an appropriate warping window length may drastically improve the performance of DTW-based methods, with verified results in the clustering task (DAU; BEGUM; KEOGH, 2016), the topic of the next section. Moreover, ensemble methods, mostly including 1-NN DTW as one of the classifiers, have presented better accuracy results than any mentioned algorithm, with the cost of a higher learning and classification runtime (BAGNALL *et al.*, 2017).

1.3.3 Clustering

The main difference between classification and clustering is related to the existence of class labels. In the clustering task, the goal is to find partitions or hierarchies to define groups (*clusters*) so that the objects in the same group are somehow related to each other. A good clustering can be seen as that one in which the objects in the same cluster are similar to each other and objects in distinct groups are dissimilar.

A practical example of this task is the text clustering ([MARCACINI; REZENDE, 2010](#)). Consider an extensive collection of scientific, news or fictional documents. One may want to read different documents of the same topic, such as news about the same event but from different points of view. Grouping documents that have a similar frequency of the same relevant words in the text is a common approach for this problem.

Clustering is also an intermediate step for different tasks. A simple example is the clustering of music excerpts to create a codeword-based representation for the automatic identification of music genre ([SILVA *et al.*, 2014](#)).

According to [Han, Kamber and Pei \(2011\)](#), the clustering algorithms can be divided into five broad categories: hierarchical, partitioning, density-based, grid-based and model-based methods. Hierarchical clustering finds a tree of groups so that each level of the tree has a different number of groups, obtained by splitting a group in two (divisive) or joining two groups in one (agglomerative). Partitioning methods, as the name suggests, find partitions in the data to group similar objects in the same partition and separate dissimilar objects in different ones. The density-based methods aim to find regions in the space where the data are densely grouped, separated for a large (sparse) margin from other groups. Grid-based methods quantize the objects to a grid-structured space. Finally, the model-based algorithms try to fit the data to a predefined model, such as probability density functions or neural networks.

An important observation about clustering algorithms is that there exist a multitude of methods that require the all-pairwise (dis)similarity matrix as input. Some examples are the partitioning methods k -medoids and spectral clustering, as well as the family of agglomerative hierarchical clustering algorithms. For the sake of simplicity and readability, the interested reader may refer to ([AGGARWAL; REDDY, 2013](#)) for a formal and complete description of clustering algorithms.

There is a lack of a comprehensive experimental comparison of (dis)similarity measures for time series clustering in the literature. [Liao \(2005\)](#) surveyed papers that performed clustering of time series in different application domains. The author summarized several papers concerning (dis)similarity measure, the clustering algorithm, the evaluation criterion, and application domain. Although the author declares that the way the similarity is estimated is an important decision, there is no recommendation about how to choose the appropriate distance measure.

More recently, [Aghabozorgi, Shirkhorshidi and Wah \(2015\)](#) studied more than two hundred papers in the related literature. The authors stated that most of the surveyed papers suffer from, among other issues, “inaccurate similarity calculation due to the high complexity of accurate measures.” In addition, they reveal that the most effective approach to calculating the distances between the time series are those based on dynamic programming to find optimal alignment path, such as DTW.

The authors also pointed to the fact that partitioning techniques and hierarchical clustering methods are the most used categories of clustering algorithms for time series data. Partitioning methods are common because several methods in this category achieve good results with a fast response. On the other hand, the need for predefining the number of clusters is a limitation of this approach. The hierarchical clustering suppresses this limitation, but are less often used because of its high computational cost. Specifically, this class of algorithm usually requires the all-pairwise distance matrix.

Despite the use of density-based methods being rare, recent work has stated that an augmented version of the method Density Peaks ([RODRIGUEZ; LAIO, 2014](#)) is an accurate alternative for time series clustering ([BEGUM *et al.*, 2015](#); [DAU; BEGUM; KEOGH, 2016](#)). All the operations of this method are based on the definition of density, given by the number of neighbors in a predefined radius. Also, the objects are associated with clusters according to their nearest neighbors with higher density. Note that, once these operations are defined regarding their nearest neighbors, techniques to speed up similarity search can significantly improve the efficiency of Density Peaks ([BEGUM *et al.*, 2015](#)).

1.3.4 Motifs and Discord Discovery

Time series motifs and discord discovery are temporal data mining tasks whose definitions are directly related to (dis)similarity measures. In an informal definition, motifs and discords are typical and atypical patterns, respectively.

Formally, the most significant motif may be defined by (at least) two points of view ([MUEEN, 2014](#)): (i) similarity-based motifs – the closest pair of subsequences in the dataset; (ii) support-based motifs – the subsequence with the highest number of neighbors. The neighborhood of support-based motifs is defined by a threshold of distance, similar to the maximum allowed distance parameter of range queries ([RAFIEI; MENDELZON, 1997](#)). In contrast, the similarity-based motifs are strictly related to the nearest neighbor search.

The definition of time series discord is the opposite of the definition (i) of motifs. A discord is a subsequence from a long time series that is maximally different to all the remaining subsequences ([KEOGH *et al.*, 2007](#)). In a practical standpoint, both primitives are based on the nearest neighbors of each subsequence. Motifs are the subsequences in which the distances to

their neighbors are minimal and discords are the subsequences whose distances to its nearest neighbor are maximal.

Motif and discord discovery are naturally costly tasks, given that is necessary to assess the neighborhood of every subsequence in a (usually) long time series. However, in general, both motif and discord discovery may be performed as a modification of similarity search. This fact allows the use of several techniques to accelerate the algorithms to perform these tasks.

In fact, the state-of-the-art method for the similarity-based motifs and discords discovery (YEY *et al.*, 2017) is based on the fastest algorithm to make similarity search under the Euclidean distance known so far (MUEEN *et al.*, 2017). Such similarity search algorithm uses the Fast Fourier Transform (WALKER, 1996) to calculate the cross-correlation, i.e., the sliding dot-product, between a short time series (query) and a long one (reference). Using this approach, the Euclidean distance between the query and each subsequence of the reference time series is obtained directly from the dot-products and the cumulative sum of the observations from these time series. This operation takes a time proportional to $\mathcal{O}(n \log n)$, instead of the $\mathcal{O}(n^2)$ complexity of a straightforward implementation.

The algorithm STAMP (YEY *et al.*, 2016) uses this similarity search method adopting each subsequence of the time series as the query. At the end of each search, we have the distance between the subsequence and the entire time series. Instead of storing all the calculated distances, STAMP only stores two pieces of information: (i) the distance of each subsequence to its nearest neighbor; and (ii) the initial position of such neighbor in the reference time series. These structures are called Matrix Profile (MP) and Matrix Profile Index (MPindex), respectively.

With the MP at hand, finding motifs and discords is straightforward. The position of the maximum value in the MP points to the first position of the subsequence considered as discord. In contrast, the minimum value (repeated at least once) points to the position of a motif. To recover the pair of motifs, we just need to check the nearest neighbor of the discovered motif in the MPindex.

The algorithm STOMP (ZHU *et al.*, 2016) is also used to calculate the MP, but is much faster than STAMP. Instead of calculating the cross-correlation for every subsequence in the time series, STOMP reuses previous calculations. For a detailed explanation on how to use dot-products on the Euclidean distance calculation, as well as how to reuse them to accelerate the motifs discovery, we refer the reader to Chapter 5.

While STAMP and STOMP were designed for the Euclidean distance, the MP and MPindex constitute a space efficient representation of the relations of subsequences which may be built for any distance measure. Also, although we have presented the MP as a representation to find motifs and discords, it is suitable for a multitude of tasks.

1.3.5 Information Retrieval

The task of information retrieval (IR) is the search for relevant information in a collection of a specific type of data (MANNING; RAGHAVAN; SCHÜTZE, 2008). Usually, a retrieval system must find a ranking of objects in the collection so that the most relevant objects, according to a query, appear as close as possible to the top of the ranking. Besides, this query does not necessarily belong to the same type than the objects to be retrieved. For instance, an image retrieval system may receive a text as input. When the user searches for “dog,” the IR system needs to return a set of pictures of dogs.

Usually, IR is not categorized as a data mining task. Actually, IR and data mining are seen to be different subfields of the computer science. While data mining aims to find *hidden* knowledge in the data, IR intends to recover unstructured data in extensive collections according to the user’s needs. In this case, the data are presented to the user in the way it is stored, which does not imply that a discovery was made (BEN-DOV; FELDMAN, 2009). However, there is a wide intersection between these areas. It can be verified, for instance, by the affinity between IR and the similarity search task. Also, data mining algorithms are frequently part of the IR procedure (MANNING; RAGHAVAN; SCHÜTZE, 2008).

Commonly, the IR is associated with text, image or multimedia data collections. In many of these scenarios, the data may include or be represented as time series. A prominent example is the music information retrieval (MIR). In this case, a user may be interested in all the songs that belong to the same genre that her/his favorite music or to find all recordings that represent versions of an original song.

This application is particularly attractive to this research since it requires the calculation of the similarity between many objects. In particular, for each query, there is the need for many or even all query-to-object distances, depending on the task specification. Also, it allows us to evaluate how the proposed methods behave for multidimensional time series. MIR systems usually transform music signals in a different time series through feature extraction (e.g. tonal features (MÜLLER; KURTH; CLAUSEN, 2005)). For this, we slide a window through the signal extracting features in consecutive moments of time, creating a new multidimensional time series to represent the original signal.

MIR uses several distance measures for different purposes. Because music data severely suffers from distortions in the time axis (mainly because of differences in tempo), it usually requires distance measures able to match the observations nonlinearly. Some examples are the use of DTW for MIR according rhythm (REN; FAN; MING, 2016) or emotion (DENG; LEUNG, 2015) and for cover song identification (FANG; DAY; CHANG, 2016).

Another application of time series in the IR task which has attracted attention in the last years is the temporal information retrieval of text data (CAMPOS *et al.*, 2015). An evident example of this approach is the construction of time series to measure the frequency in which

a term of interest appears over time. This is a common approach to extract knowledge from social media, such as topics from Twitter data (THELWALL, 2014). Another application is the time series forecasting using “external time series” derived from textual news (MARCACINI; CARNEVALI; DOMINGOS, 2016). The main idea of this application is the refinement of forecasting models based on similar events occurred in the past. This is done by matching time series of term frequency in related news articles.

1.4 Main Contributions

The general objective of this work is to develop novel similarity-based data mining algorithms for time series which are efficient concerning time and space and methods able to improve the scalability of existing algorithms.

Note that this goal does not limit our research to any specific distance measure or data mining task. However, the stated quality of DTW in addition to its costly algorithm suggests the need of investigation on this distance measure. At the same time, mining tasks solved by costly methods, such as motifs discovery, are also taken as an interesting topic of research.

With this in mind, this work diversified the investigated task and similarity measure. Consequently, the contributions of this thesis are also diversified. The main contributions of this research may be summarized as follows.

- We studied how an unnoticed or underappreciated consequence of the segmentation of streaming time series affects the temporal mining and proposed an algorithm to provide invariance to this effect;
- We developed an algorithm to speed up the exact calculation of DTW, which can be applied in any similarity-based time series mining task;
- Embedding this algorithm in the similarity search procedure, we could improve the efficiency of the fastest tool for nearest neighbor search under the DTW distance known so far;
- We developed a novel efficient Euclidean distance-based method to assess similarity in music data with implications on information retrieval, visualization, and other tasks;
- We presented a method to find motifs and discords with different lengths and local scaling in time series data in a feasible time. This method is based on a novel modification of the DTW and is anytime, i.e., may be interrupted at any step of its execution returning a good approximate solution.

The next section describes this thesis’ organization, in which we split the pointed contributions into different chapters. Chapter 7 briefly describes other contributions of this work.

1.5 Thesis Organization

This thesis is organized as a collection of papers. For this, the papers or submitted manuscripts which better summarize the contributions of this work are presented in the format of chapters (namely, Chapters 2 to 6). The text and figures of these chapters are the same that in the papers, except for minor modifications to adapt to the thesis format. The only exception is the Chapter 2, which is an extended version of the published paper. Specifically, that paper was published in a short version, but we present it in the original format, i.e., as a regular paper.

For this reason, each of these chapters is self-contained. In other words, each chapter contains all the necessary background, the required definitions, and the used notation. Consequently, the reader may feel free to read these chapters in any preferred order.

In the remaining of this section, we introduce the contents of each chapter of this thesis.

1.5.1 Chapter 2

Title: "*Prefix and Suffix Invariant Dynamic Time Warping*." This chapter is an extended version of the paper published in the 2016 IEEE International Conference on Data Mining (SILVA; BATISTA; KEOGH, 2016).

As presented in the Section 1.1.2, the performance of some temporal data mining methods may depend on the invariances provided by the distance measure. Namely, some simple pre-processing and subtle modifications of the distance calculation may aggregate invariance to amplitude, offset, occlusion, complexity, among others (BATISTA *et al.*, 2014). In this work, we discuss an effect that seemed to be unnoticed by the time series community, as well as we present a novel method to provide invariance to it.

While many time series benchmark datasets guarantee a perfect segmentation of subsequences, most practical problems on time series mining require the automatic segmentation of time series in a streaming fashion. In this case, the presence of observations in the segmented subsequences which do not belong to the event of interest is virtually unavoidable. We name such spurious data points as prefix and suffix, depending on the positions of these values in the subsequences.

In Chapter 2, we discuss the effects of these undesirable observations in the endpoints on the classification of time series. Then, we present the Prefix and Suffix Invariant Dynamic Time Warping (ψ -DTW), a novel algorithm to deal with endpoint variances.

In summary, the main contributions of this work are:

- We discuss an unnoticed relevant issue on time series similarity comparison and present a solution for it;
- Our method is simple and easy to implement;

- The classification results using our algorithm are significantly better than the accuracy obtained by DTW;
- We present an efficient and tight lower bound for our distance measure, allowing the use of the widely known indexing techniques.

It is important to notice that we also used ψ -DTW in other tasks beyond classification. Chapter 6 presents its application in motifs and discord discovery.

1.5.2 Chapter 3

Title: "*Speeding Up All-Pairwise Dynamic Time Warping Matrix Calculation.*" This chapter is based on the paper published in the 2016 SIAM International Conference on Data Mining (SILVA; BATISTA, 2016b).

The main issue of using Dynamic Time Warping on time series mining is its quadratic complexity. While the literature presents methods to significantly speed up the similarity search under DTW (RAKTHANMANON *et al.*, 2012), some data mining tasks require the all-pairwise distance matrix between the time series objects. The techniques used to accelerate the similarity search are not applicable in this scenario. Specifically, these techniques are used to avoid the DTW calculation between most of the pairs of subsequences. When we are required to have all (or most of) the distances between the pairs of objects, these techniques are not suitable.

For this reason, several authors proposed approximations of the DTW distance. However, these approximations do not guarantee any bounds for the error.

We present the first exact method to speed up the DTW calculation. Our method prunes cells of the dynamic programming matrix used to calculate DTW that are guaranteed to not lead to the optimal solution.

The main advantages of our algorithm, named Dynamic Time Warping with Pruned Warping paths (PrunedDTW), are:

- It is the first algorithm able to speed up the exact DTW calculation with no external information, such as the obtained by indexing techniques, supporting both warping-constrained and unconstrained versions of the distance calculation;
- The improvement of our method is proportional to the time cost of DTW. In other words, our method is more effective in scenarios where DTW performs worst;
- Our method is a general framework which may be improved with the proposal of better upper bounds for DTW.

1.5.3 Chapter 4

Title: “*Speeding Up Similarity Search Under Dynamic Time Warping by Pruning Unpromising Alignments.*” This chapter is based on a manuscript by Diego F. Silva, Rafael Giusti, Gustavo E. A. P. A. Batista, and Eamonn Keogh submitted to the Data Mining and Knowledge Discovery journal.

Similarity search is probably the most common time series mining task. In addition to being widely used in the exploratory data analysis, it is often an intermediate step in several other tasks, such as classification, clustering and motif discovery. At the same time, there is a large body of work demonstrating the utility of the DTW in the similarity search.

Given that the algorithm to calculate DTW is quadratic, its use to assess a high volume of data is impractical. For this reason, the research community spent a tremendous effort in the last decades to index the similarity search under DTW, i.e., to create techniques to avoid the calculation of the distances from the current query to most of the subsequences in the database (KIM; PARK; CHU, 2001; FALOUTSOS; RANGANATHAN; MANOLOPOULOS, 1994; KEOGH; RATANAMAHATANA, 2005; RAKTHANMANON *et al.*, 2012). In another point of view, these techniques are methods to reduce the search space.

In this work, we investigate the answer for the following questions. “What is the (current) worst case of time series similarity search under DTW?” “Is there a clear bottleneck in this case?” “How to improve the efficiency of the similarity search in the worst case?”

The efficiency of the similarity search under DTW depends on three key features: (i) the length of the time series; (ii) the length of the query; and (iii) the warping window length. For each of these features, the larger it is, the slower is the search. In addition, we found that the bottleneck of the search is not the indexing phase, but the calculation of the DTW in the cases it is necessary, even if the number of pairs to be compared is significantly small in comparison to the length of the time series.

For these reasons, we propose the use of PrunedDTW as a technique to improve the worst case of the similarity search under DTW. The PrunedDTW has a characteristic that makes its use intuitive in these cases: usually, the larger the subsequences under comparison and/or the length of the warping window, the better the improvement provided by the PrunedDTW.

In summary, the main contributions of this work are:

- Our suite of techniques is a subtle modification of the widely used UCR Suite;
- We adapt the PrunedDTW to the similarity search task by using information obtained during the search to make it even faster;

- We compared our results to the state-of-the-art tool for similarity search and showed that our method performs similarly in the best case and much more fastly in the worst case of the search;
- Given that the gain obtained by our method is more expressive for long queries and/or large warping windows, we discuss scenarios that require these characteristics.

1.5.4 Chapter 5

Title: “*Fast Similarity Matrix Profile for Music Analysis and Exploration.*” This chapter is based on a manuscript by Diego F. Silva, Chin-Chia M. Yeh, Yan Zhu, Gustavo E. A. P. A. Batista, and Eamonn Keogh, submitted to the IEEE Transactions on Multimedia. The submitted manuscript is an extended version of the paper published in the 2016 International Society for Music Information Retrieval Conference ([SILVA et al., 2016](#)).

The Matrix Profile (MP) is a novel representation of the distances between subsequences of a time series or between two distinct temporal data ([YEH et al., 2016](#)). This representation is a space efficient way to keep the distance between each subsequence and its nearest neighbor, as well as a structure to store which subsequence is the nearest neighbor related to such a distance.

The MP is the state-of-the-art representation to solve several problems in temporal data mining under the Euclidean Distance (ED). One such example is the motif and discord discovery. In addition to being simple and space efficient, the MP is calculated by the fastest algorithm for similarity search under ED know so far.

The work presented in this chapter is the proposal of using the MP for multidimensional time series to deal with music data. Music data are audio signals usually assessed by extracting features in a sliding window fashion. As a result, these features can be seen as multidimensional time series.

In general, the distance (or similarity) information on music data is assessed by using the whole distance matrix, which is space and time inefficient. We show that the proposed Similarity Matrix Profile (SiMPle) is space and time efficient and present examples of how to use it in several distinct tasks. Also, we make our method fast by adapting the algorithm STOMP ([ZHU et al., 2016](#)), which reuses previous computation to speed up dot-product calculations.

In summary, the contributions of this work are:

- The proposed method is fast and space efficient;
- We extend the recently proposed Matrix Profile representation to the multidimensional case and apply it in music data mining, retrieval, and visualization tasks;
- We focus on the cover song identification task, demonstrating that our method suppresses other similarity-based algorithms.

1.5.5 Chapter 6

Title: “*Elastic Time Series Motifs and Discords.*” This chapter is based on a manuscript by Diego F. Silva and Gustavo E. A. P. A. Batista, submitted to the 2017 IEEE International Conference on Data Mining.

Motifs discovery is a widely studied problem on the time series mining literature (MUEEN, 2014; TORKAMANI; LOHWEG, 2017). At the same time, the introduction of the Matrix Profile demonstrates that both motifs and discords can be calculated in the same procedure. However, the majority of work on motifs and discords discovery is based on the Euclidean Distance.

Although the Dynamic Time Warping (and its variants) is the most relevant distance measure for time series mining, it is not commonly used for motifs discovery. The main reason for this is the efficiency-related issues.

In the work presented in Chapter 6, we propose a set of techniques based on the UCR Suite (RAKTHANMANON *et al.*, 2012) for the fast calculation of the Matrix Profile based on the ψ -DTW for motifs and discords discovery. Also, we discuss the motivation and consequences of using the ψ -DTW instead the traditional DTW in this task.

The main contributions of this work are:

- We introduce the ψ -DTW for the motifs and discords discovery and demonstrate its utility in these tasks;
- As consequence of using ψ -DTW, our method allows discovering motifs with slightly different lengths with no additional steps in the algorithm to ensure that;
- We present a suite of techniques to perform the motifs and discords discovery under ψ -DTW in a feasible time;
- Our method is anytime, i.e., can be interrupted at any step of the algorithm’s execution, and we demonstrate that the best motifs can be found in less the 5% of the runtime to calculate the full MP.

1.5.6 Chapter 7

The results obtained during the period of this research go beyond the contributions presented in the previous sections. This thesis does not highlight these results for the sake of cohesion between the presented ideas and contributions.

Another reason to not present a whole chapter for some of the produced research papers is the participation as co-author in such publications, instead of being the first author. Specifically, we had established higher priority to the papers in which the author of this thesis is the first author of – and, consequently, the main responsible for – the paper.

In this chapter, we briefly present other publications originated from this work. Specifically, we list all the papers produced during the period of this project. These publications are split on time series classification, streaming data mining, and music data mining and retrieval.

1.5.7 Chapter 8

Chapter 8 concludes this work by making some general considerations of the presented contributions. Also, it points to open related problems and, consequently, some directions for future work.

PREFIX AND SUFFIX INVARIANT DYNAMIC TIME WARPING

Abstract: While there exist a plethora of classification algorithms for most data types, there is an increasing acceptance that the unique properties of time series mean that the combination of nearest neighbor classifiers and Dynamic Time Warping (DTW) is very competitive across a host of domains, from medicine to astronomy to environmental sensors. While there has been significant progress in improving the efficiency and effectiveness of DTW in recent years, in this work we demonstrate that an underappreciated issue can significantly degrade the accuracy of DTW in real-world deployments. This issue has probably escaped the attention of the very active time series research community because of its reliance on static highly contrived benchmark datasets, rather than real world dynamic datasets where the problem tends to manifest itself. In essence, the issue is that DTW’s eponymous invariance to warping is only true for the main “body” of the two time series being compared. However, for the “head” and “tail” of the time series, the DTW algorithm affords no warping invariance. The effect of this is that tiny differences at the beginning or end of the time series (which may be either consequential or simply the result of poor “cropping”) will tend to contribute disproportionately to the estimated similarity, producing incorrect classifications. In this work, we show that this effect is real, and reduces the performance of the algorithm. We further show that we can fix the issue with a subtle redesign of the DTW algorithm, and that we can learn an appropriate setting for the extra parameter we introduced. We further demonstrate that our generalization is amiable to all the optimizations that make DTW tractable for large datasets.

2.1 Introduction

Following the huge growth of applications based on temporal measurements, such as Quantified Self and Internet of Things (SWAN, 2012), time series data are becoming ubiquitous

even in our quotidian lives. It is increasingly difficult to think of a human interest or endeavor, from medicine to astronomy, that does not produce copious amounts of time series.

Among all the time series mining tasks, query-by-content is the most basic. It is the fundamental subroutine used to support nearest-neighbor classification, clustering, etc. The last decade has seen mounting empirical evidence that the unique properties of time series mean that Dynamic Time Warping (DTW) is the best distance measure for time series across virtually all domains, from activity recognition for dogs (KIYOHARA *et al.*, 2015) to classifying star light curves to ascertain the existence of exoplanets (DEBRAY; WU, 2013).

However, virtually all current research efforts assume a perfect segmentation of the time series. This assumption is engendered by the availability of dozens of contrived datasets from the UCR time series archive (CHEN *et al.*,). Improvements on this (admittedly very useful) resource have been seen as sufficient to warrant publication of a new idea, but it would be better to see success on these benchmarks as being only necessary to warrant consideration of a new approach.

In particular, the way in which the majority of the datasets were created and “cleaned” means that algorithms that do well on these datasets can still fail when applied to real world streaming data. The issue lends itself to a visually intuitive explanation. Figure 7 shows two examples from the Australian Sign Language dataset aligned by DTW. We can see the utility of DTW here, as it aligns the later peak of the blue (bold) time series to the earlier occurring peak in the red (fine) time series. However, this figure also illustrates a weakness of DTW. Because every point must match another one, the first few points in the red sequence are forced to match the first point in the blue sequence.

Figure 7 – *top*) Two time series compared with DTW. While the prefix of the red (fine) time series consists of only 6% of the length, it is responsible for 70.5% of the error. *bottom*) We propose to address this disproportionate appointing of error by selectively ignoring parts of the prefix (and/or suffix)

Source: Elaborated by the author.

While Figure 7 does show the problem on a real data object, the reader may wonder how common this issue is “in the wild”. We claim that at least in some domains, this problem is very common.

For example, heartbeat extraction algorithms often segment the signal to begin at the maximum of the QRS complex (SUHRBIER *et al.*, 2006). However, as shown in Figure 8, this location has the greatest variability in its prefixes and suffixes.

Figure 8 – Three heartbeats taken from a one-minute period of a healthy male. The beats were extracted by a state-of-the-art beat extraction algorithm (SAINI; SINGH; KHOSLA, 2013), but there is significant variation in the prefix (all three) and in the suffix (green vs. the other two)

Source – Elaborated by Dr. Eamonn Keogh

Similar remarks apply to gait cycle extraction algorithms (TABORRI *et al.*, 2016). Likewise, star light curves, for which DTW is known to be very effective, have cycles extracted by a technique called universal phasing (REBBAPRAGADA *et al.*, 2009). However, universal phasing has the unfortunate side effect of placing the maximum variance at the prefix and suffix of the signals

In this work, we address this problem of uninformative and undesirable “information” contained just before and just after the temporal measurement of informative data. For the sake of clarity, we will refer to these unwanted values as *prefix* and *suffix*, and use endpoints to refer to both.

Our approach is simple and intuitive, but highly effective. We modify the endpoint constraint of Dynamic Time Warping (DTW) to provide endpoint invariance. The main idea behind our proposal is allowing DTW to ignore some leading/trailing values in one or both of the two time series under comparison. While our idea is simple, it must be carefully executed. It is clear that ignoring too much (useful) data is just as undesirable as paying attention to spurious data.

We note that somewhat similar observations were known to the signal processing community when DTW was the state-of-the-art technique for speech processing (in the 1980’s and 90’s before being superseded by Markov models (RABINER; JUANG, 1993)). However, the importance of endpoint invariance for time series seems to be largely unknown or underappreciated (RATANAMAHATANA; KEOGH, 2005; KEOGH; RATANAMAHATANA, 2005; RAKTHANMANON *et al.*, 2012).

We can summarize the main contributions of this paper as follows¹:

- We draw the data mining community’s attention to the endpoint invariance, which seems to be a little or no considered issue;
- We propose a modification of the well-known algorithm Dynamic Time Warping to provide invariance to endpoints;
- Although simple and intuitive, we show that our method can considerably improve the classification accuracy when warranted, and just as importantly, our ideas do not reduce classification accuracy if the dataset happens to not need endpoint invariance;
- Unlike other potential fixes, our distance measure respects the property of symmetry and, consequently, can be applied in a multitude of data mining algorithms with no pathological errors caused by the order of the data;
- In spite of the fact that we must add a parameter to DTW, we show that it is possible to robustly learn a good value for this parameter using only the training data.

2.2 Time Series Suffix and Prefix

Most research efforts for time series classification assume that all the time series in the training and test sets are carefully segmented by using the precise endpoints of the desirable event ([RATANAMAHATANA; KEOGH, 2005](#); [REBBAPRAGADA *et al.*, 2009](#); [UENO *et al.*, 2006](#); [WANG *et al.*, 2013](#)). Despite the ubiquity of time series datasets that fulfill such an assumption, in practical situations the exact endpoints of events are difficult to detect. In general, a perfectly segmented dataset can only be achieved by manual segmentation or some contrivance that uses external information.

To see this, we revisit the Gun-Point dataset, which has been used in more than two hundred papers to test the accuracy of time series classification ([CHEN *et al.*,](#)). As shown in Figure 9, the data objects considered in such a set have perfectly flat prefixes and suffixes. However, these were obtained only by carefully prompting the actor’s movements (pointing a gun or a finger) with a metronome that produced an audible cue every five seconds.

In more realistic scenarios, the event of pointing a gun/finger must be detected among several different movements. Before drawing the weapon, the actor could be running, talking on a cell phone, etc.

For example, consider the scenario in which some movement was performed just before the weapon was aimed. In addition, another movement started immediately after the gun was

¹ In this paper, we experiment our method in the classification scenario. Chapter 6 of this thesis presents the results of our proposal for the motif and discord discovery tasks.

Figure 9 – The ubiquitous Gun-Point dataset was created by tracking the hand of an actor (*top*). However, the perfectly flat prefix and suffix were due to carefully training the actor to have her hand immobile by her side one second before and one second after the cue from a metronome (*bottom*)

returned to the holster. In this case, the time series could have a more complex shape as shown in Figure 10. As visually explained in Figure 7, it is clear that prefix and suffix would greatly prejudice the distance estimation in this case.

Figure 10 – Example of a time series containing the event to be classified (*blue*) with prefix and suffix information (*red*)

Another possible issue that can result from automatic segmentation is the algorithm used to extract the time series be too “aggressive” and make the mistake of truncating the last few observations of the event of interest. Obviously, a similar issue could also happen at the beginning of the signal.

In this case, the time series is missing its true suffix. Even with such missing information, the shape that describes the beginning of the action may be enough such that it will be classified correctly. However, the object that would otherwise be considered its nearest neighbor may contain information of the entire movement, as shown in Figure 10. To classify this kind of badly cropped item correctly, a distance measure must avoid matching the last few observations of the complete event to the values observed in our badly segmented event. In Section 2.5 we will show how our method can solve these issues.

2.3 Definitions and Background

A time series x is a sequence of n ordered values such that $x = (x_1, x_2, \dots, x_n)$ and $x_t \in \mathbb{R}$ for any $t \in [1, n]$. We assume that two consecutive values are equally spaced in time or the interval between them can be disregarded without loss of generality. For clarity, we refer each value x_t as observation.

The Dynamic Time Warping (DTW) algorithm is arguably the most useful distance measure for time series analysis. For example, mounting empirical evidence strongly suggest that the simple nearest neighbor algorithm using DTW outperforms more “sophisticated” time series classification methods in a wide range of application domains ([WANG et al., 2013](#)).

In contrast to other distance measures, such as those in the L_p -norm family, the DTW computes a non-linear alignment between the observations of the two time series being compared. In other words, while L_p -norm distances are only able to compare the value x_t to a value y_t of a time series y , DTW is able to compare x_t to y_s such that $t \cong s$.

To compute the optimal non-linear alignment between a pair of time series x and y , with lengths n and m respectively, the DTW typically bound to the following constraints:

- **Endpoint constraint.** The matching is made for the entire length of time series x and y . Therefore, it starts at the pair of observations $(1, 1)$ and ends at (n, m) ;
- **Monotonicity constraint.** The relative order of observations must be preserved, i.e., if $s_1 < s_2$, the matching of x_t with y_{s_1} is done before matching x_t with y_{s_2} ;
- **Continuity constraint** The matching is made in one-unit steps. It means that the matching never “jumps” one or more observations of any time series.

The calculation of DTW distance is performed by a dynamic programming algorithm. The initial condition of such an algorithm is defined by Equation 2.1.

$$dtw(i, j) = \begin{cases} \infty, & \text{if } i = 0 \text{ or } j = 0 \\ 0, & \text{if } i = j = 0 \end{cases} \quad (2.1)$$

In order to find the optimal non-linear alignment between the observations of the time series x and y , DTW follows the recurrence relation defined by Equation 2.2.

$$dtw(i, j) = c(x_i, y_j) + \min \begin{cases} dtw(i-1, j) \\ dtw(i, j-1) \\ dtw(i-1, j-1) \end{cases} \quad (2.2)$$

where $i \in [1, n]$ and $j \in [1, m]$, m being the length of the time series y . The partial $c(x_i, y_j)$ represents the cost of matching two observations x_i and y_j and is calculated by the squared Euclidean distance between them. Finally, the DTW distance returned is $DTW(x, y) = dtw(n, m)$.

An additional constraint commonly applied to DTW is the warping constraint. This constraint limits the time difference that the algorithm is allowed to match the observations. In the matrix view of DTW, this constraint limits the algorithm to calculate the values of the DTW matrix in a region close to its main diagonal. The benefit of using a warping constraint is two fold: the DTW calculation takes less time (as it is not necessary to calculate values for the entire distance matrix) and it avoids pathological alignments. For example, when comparing heartbeats, we want to allow a little warping flexibility to be invariant to small (and medically irrelevant) changes in timing. However, it never makes sense to attempt to align ten heartbeats to twenty-five heartbeats. The warping constraint prevents such degenerate solutions. As a practical confirmation of its utility using the constraint, we note that it has been shown to improve classification accuracy (RATANAMAHATANA; KEOGH, 2005).

The most common warping constraint for DTW is the Sakoe-Chiba warping window (SAKOE; CHIBA, 1978). The use of warping constraints adds a parameter to be set by the user. However, several studies show that small windows (usually smaller than 10%) are usually a good choice for nearest neighbor classification (RATANAMAHATANA; KEOGH, 2005).

2.4 Related Work

The utility of relaxing the endpoint constraint of DTW has been previously noticed by the signal processing community, in the context of speech (HALTSONEN, 1984) and music analysis (MYERS; RABINER, 1981). However, the issue seems to be unknown or glossed over in time series data mining.

The time series mining method that shares more similarities to our proposal is the open-end DTW (OE-DTW) (TORMENE *et al.*, 2009). However, OE-DTW was proposed to match incomplete time series to complete references. In other words, such a method is based on the assumption that we can construct a training set with carefully cropped time series and we can know the exact point that represents the beginning of the time series to be classified.

Specifically, OE-DTW is a method that allows ignoring any amount of observations at the end of the training time series. The final distance estimate is the value represented by $\min_{0 \leq i \leq m} DTW(n, i)$, i.e., the final distance is the minimum value in the last column of the DTW matrix.

A weakness of OE-DTW is that it does not consider the existence of prefix information. A modification of the OE-DTW called open-begin-end DTW (OBE-DTW) or subsequence

DTW (MÜLLER, 2007) mitigates this issue. OBE-DTW allows the match of observations to start at any position of the training time series. To allow DTW to do this, the algorithm needs to initialize the entire first column of the DTW matrix with zeros.

Although OBE-DTW recognizes that both prefix and suffix issues may exist, it only addresses the problem in the training time series. A more important observation is that OBE-DTW is not symmetric, which severely affects its utility. For example, the results obtained by OBE-DTW in any clustering algorithm are dependent on the order in which the algorithm processes the time series. To see this, consider the hierarchical single-linkage clustering algorithm (XU; WUNSCH, 2008). Figure 11 shows the result of clustering the same set of five time series objects from the Motor Current dataset (c.f. Section 2.6), presented in different orders to the clustering algorithm. Specifically, the distance between the time series x and y is calculated by $OBE DTW(x, y)$ in the first case and by $OBE DTW(y, x)$ in the second. Note that the results are completely different, a very undesirable outcome.

Figure 11 – Clustering results of the same dataset by using OBE-DTW. The difference between the results is given by the fact that they were obtained by presenting the time series in a different order to the clustering algorithm

Source: Elaborated by the author.

In addition to this issue, OBE-DTW has one other fatal flaw. In essence, it can be “too invariant,” potentially causing meaningless alignments in some cases. Figure 12 shows an extreme example of this. In the top figure, all observations of flat line match to a single observation in the sine wave, and the DTW distance obtained is 0.07. In the bottom figure, we reverse the roles of reference and query. This time, all observations of the sine wave match to a single observation in the flat line, and the DTW distance obtained is 69.0. We observe a three orders of magnitude difference in the DTW results.

Similar to the OBE-DTW, the method proposed in this paper is based on a relaxation of the endpoint constraint. However, our method is symmetric and strictly limits the amount of the signals that can be ignored, preventing the meaningless alignments shown in Figure 12. Figure 13 shows a comparison of the results obtained by the classic DTW, the OBE-DTW, and the distance measure proposed in this work when used to cluster the time series data considered in Figure 11.

Figure 12 – The OBE-DTW alignment for the same pair of time series. In the first (*top*), a sine wave is used as reference and the flat line is used as query. In the second (*bottom*), the same sine wave is used as query while the flat line is used as reference

Source: Elaborated by the author.

Figure 13 – Clusterings on a toy dataset using the classic DTW (*left*), OBE-DTW (*center*), and the distance measure proposed in this paper (*right*). Note that our method achieves a perfect and intuitive separation of the different classes

Source: Elaborated by the author.

Finally, some other algorithms were proposed to ignore the matching of some observations, but in a different scenario. Specifically, some algorithms were proposed to avoid errors of measurements, noise, occlusion, among other problems that may occur along the time series. Usually, such methods require a gap penalty or a threshold to decide whether two points of the time series should be considered a match or not. This is the case, for instance, of the Longest Common Subsequence (LCSS) [12].

2.5 Prefix and Suffix-Invariant DTW (ψ -DTW)

While there are many different methods proposed for time series classification (decision trees, etc.), it is known that the simple nearest neighbor is extremely competitive in a wide range of applications and conditions (WANG *et al.*, 2013). Given this, the only decision left to the user is the choice of the distance measure.

In most cases, this choice is guided by the invariances required by the task and domain (BATISTA *et al.*, 2014). In conjunction with simple techniques, such as z-normalization, DTW can provide several invariances like amplitude, offset and the warping (or local scaling) itself.

In this work, we address what we feel is the “missing invariance,” the invariance to spurious prefix and suffix information. Given the nature of our proposal, we call our method Prefix and Suffix-Invariant DTW, or simply PSI-DTW (or ψ -DTW for short).

The relaxed version of the endpoint constraint proposed in this work is defined as the following.

Definition Relaxed endpoint constraint: Given an integer value r , the alignment path between the time series x and y starts at any pair of observations in $\{(1, c_1 + 1)\} \cup \{(c_1 + 1, 1)\}$ and ends at any pair in $\{(n - c_2, m)\} \cup \{(n, m - c_2)\}$, such that $c_1, c_2 \in [0, r]$.

This relaxation of the endpoint constraint can avoid undesirable matches at the beginning and the end of any x or y time series by removing the obligation for the alignment path to start and end with specific pairs of observation, namely the first and the last pairs. The value r used in this definition is the relaxation factor parameter that needs to be defined by the user.

We recognize the general undesirability of adding a new parameter to an algorithm. However, we argue it is necessary (c.f. Section 2.4). In addition, we show that we are able to learn an appropriate r solely from the training data. We will return to this topic in Section 2.6.

An important aspect of the proposed endpoint constraint is the fact that, by definition, the same number of cells is “relaxed” for both column and row in the cumulative cost matrix. This is what guarantees the symmetry of ψ -DTW. If the number of relaxed columns and rows was different, the starting and finishing cells of the alignment found by ψ -DTW(x, y) could be outside of the region defined by the endpoint constraint in the cost matrix used by ψ -DTW(y, x).

The relaxation of endpoints slightly affects the initialization of the DTW estimation algorithm defined in Equation 2.1. To accomplish the new constraint, the initialization of DTW needs to be changed to Equation 2.3.

$$dtw(i, j) = \begin{cases} \infty, & \text{if } (i = 0 \text{ and } j > r) \text{ or } (j = 0 \text{ and } i > r) \\ 0, & \text{if } (i = 0 \text{ and } j \leq r) \text{ or } (j = 0 \text{ and } i \leq r) \end{cases} \quad (2.3)$$

In order to find the optimal non-linear alignment between the time series x and y , ψ -DTW follows the recurrence relation defined by Equation 2.4. Note that the recurrence relation is exactly the same than the classical DTW.

$$dtw(i, j) = c(x_i, y_j) + \min \begin{cases} dtw(i-1, j) \\ dtw(i-1, j-1) \\ dtw(i, j-1) \end{cases} \quad (2.4)$$

where $i \in [1, n]$ and $j \in [1, m]$, being m and n the length of the time series x and y , respectively.

Finally, the ultimate distance estimate can be directly obtained by the definition of the proposed relaxed endpoint constraint. Formally, the final distance calculation is given by Equation 2.5.

$$\begin{aligned} \psi - DTW(x, y, r) &= \min_{(i,j) \in finalSet} [dtw(i, j)], \\ finalSet &= \{(n - c_1, m)\} \cup \{(n, m - c_2)\} \forall c_1, c_2 \in [0, r]. \end{aligned} \quad (2.5)$$

The algorithm to calculate ψ -DTW is a subtle modification of the original DTW algorithm. For concreteness, Algorithm 3 describes the method in detail.

Algorithm 3 – ψ -DTW algorithm

Requer: Two user provided time series, x and y and the relaxation factor parameter r

Assegure: The ψ -DTW distance between x and y

```

1:  $n \leftarrow \text{length}(x)$ 
2:  $m \leftarrow \text{length}(y)$ 
3:  $M \leftarrow \text{infinity\_matrix}(n + 1, m + 1)$ 
4:  $M[[0..r], 0] \leftarrow 0$  ▷ where  $[0..r]$  is  $(0, 1, \dots, r)$ 
5:  $M[0, [0..r]] \leftarrow 0$ 
6: para  $i \leftarrow 1$  até  $n$  faça
7:   para  $j \leftarrow 1$  até  $m$  faça
8:      $M[i, j] = sqED(x_i, y_j) + \min(M[i-1, j-1], M[i-1, j], M[i, j-1])$ 
9:   fim para
10: fim para
11:  $minX \leftarrow \min(M[[n-r, n], m])$ 
12:  $minY \leftarrow \min(M[n, [m-r, m]])$ 
   retorna  $\min(minX, minY)$ 

```

The algorithm starts by defining the variables used to access the length of time series (lines 1 and 2) and the DTW matrix according to Equation 2.3 (lines 4 to 5). The for loops (lines 6 to 9) fill the matrix according to the recurrence relation defined in Equation 2.4. Finally, the algorithm finds the minimum value in the region defined by the new endpoint constrained (lines 11 and 12) and returns it as the distance estimate. To implement the constrained warping version

of this algorithm, one only needs to modify the interval of the second for loop (line 7) according to the constraint definition.

Note that the proposed method is a generalization of DTW, thus it is possible to obtain the classic DTW by our method. Specifically, if $r = 0$, the final result of our algorithm is exactly the same as the classic DTW.

2.6 Experimental Evaluation

We begin this section by reviewing our experimental philosophy. We are committed to reproducibility, thus we have made available all the source code, datasets, detailed results and additional experiments in a companion website for this work (SILVA; BATISTA; KEOGH, 2016). In addition to reproducing our experiments, the interested reader can use our code on their own datasets. We implemented all our ideas in Matlab, as it is ubiquitous in the data mining community.

To test the robustness of our method, we compare its performance against the accuracy obtained by the classic, unconstrained DTW. In addition, we present results obtained using constrained-warping. We refer to the constrained versions of the algorithms with names containing the letter c. For clarity, cDTW refers to the DTW with warping constraint and ψ -cDTW stands for the constrained version of ψ -DTW.

We are not directly interested in studying the effect of warping window width on classification accuracy. The value of the warping window width parameter has been shown to greatly affect accuracy, but it has also been shown to be easy to learn a good setting for this parameter with cross validation (RATANAMAHATANA; KEOGH, 2005; UENO *et al.*, 2006; WANG *et al.*, 2013). For simplicity, we fixed it as 10% of the length of the query time series by default.

However, this setting limits the choice of the relaxation factor to ψ -DTW. For any relaxation factor that is greater than or equal to the warping length, the distance is the same. For this reason, when we wanted to test the effect of larger relaxation factors, the warping window used in the experiment was set by the same value as r .

We divide our experimental evaluation into two sections.

- In order to clearly demonstrate that our algorithm is doing what we claim it can, we take perfectly cropped time series data and add increasing amounts of spurious endpoint data. This experiment simulates the scenario in which the segmentation of time series is not perfect, i.e., there are endpoints that may represent random behaviors;
- The experiments above will be telling, but unless real datasets have the spurious endpoint problem, they will be of little interest to the community. Thus, we apply ψ -DTW on real datasets that we suspect have a high probability of the presence of spurious endpoints.

For clarity of presentation, we have confined this work to the single dimensional case. However, our proposal can be easily generalized to multidimensional data.

2.6.1 The Effect of ψ -DTW on Different Lengths of Endpoints

As noted above, the UCR Time Series Archive has been useful to the community working on time series classification (CHEN *et al.*,). However, in general, the highly contrived procedures used to collect and/or clean most of the datasets prevent the appearance of prefixes and suffixes (recall Figure 9). For this reason, the impact of endpoints cannot be directly evaluated by the use of such datasets.

However, such “endpoint-free” data create a perfect starting point to understand how different amounts of uninformative data can affect both DTW and ψ -DTW. To see this, we consider some datasets that are almost certainly free of specious prefix or suffix information. To these we prepend and postpend random walk subsequences with length varying from 0% to 50% of the original data. Next, we compared the accuracy obtained using the nearest neighbor classification on the modified datasets using both DTW and ψ -DTW. At each length of added data, we average over three runs with newly created data.

At this point, we are not learning the parameter r . Instead, we fixed both the relaxation factor and warping constraint length as 10% of the time series being compared.

Intuitively, as we add more and more spurious data, we expect to see greater and greater decreases in accuracy. However, we expect that ψ -DTW degrades slower. In fact, this is the exact behavior observed in our experiments. Figure 14 shows the results on the Cricket X dataset.

Figure 14 – The accuracy after padding the Cricket X dataset with increasing lengths of random walk data. When no such spurious data is added, the accuracy obtained by the classic DTW is very slightly better. As we encounter increasing amounts of spurious data, ψ -DTW and ψ -cDTW degrade less than DTW and cDTW

Source: Elaborated by the author.

For brevity, here we show the results on only one dataset. However, we note that this result describes the general behavior of the results obtained in other datasets.

2.6.2 Case Studies

In the previous experiment, we showed the robustness of ψ -DTW in the presence of spurious prefix and suffix information in artificially contrived time series data. In this section, we evaluate our method on real data.

The datasets we consider were extracted in a scenario in which we do not have perfect knowledge or control over the events' endpoints. In some cases, the original datasets were obtained by recording sessions similar to the Gun-Point dataset (c.f. Section 2.2), in which the invariance to endpoints is enforced by the data collection procedure. In this case, we model the real world conditions by ignoring the external cues or annotations. In particular, we simulated a randomly-ordered stream of events followed by a classic subsequence extraction step. For this phase, we considered the simple sliding window approach. For additional details on the extraction phase, please refer to (SILVA; BATISTA; KEOGH, 2016).

In keeping with common practice, we adopted the use of dictionaries as training data. A data dictionary is a subset of the original training set containing only its most relevant examples. The utility of creating dictionaries is two-fold (HU; CHEN; KEOGH, 2013): it makes the classifier faster and the accuracy obtained by dictionaries is typically better than that obtained by using all the data, which may contain outliers or mislabeled data.

To compute the relevance of training examples to the classification task, we used the SimpleRank function (UENO *et al.*, 2006). This function returns a ranking of exemplars according to their estimated contribution to the classification accuracy. Then, we selected the top- k time series of each class in the dictionary, with k empirically discovered for each dataset.

The main intuition behind SimpleRank is to define a score for each exemplar based on its "neighborhood." For each exemplar t_j , its nearest neighbor s is "rewarded" if it belongs to the same class, i.e., s is used to correctly classify t_j . Otherwise, s is "penalized" by having its score decreased. Equation 2.6 formally defines the SimpleRank function.

$$rank(s) = \sum_j \begin{cases} 1, & \text{if } class(s) = class(t_j) \\ -\frac{2}{\#classes-1}, & \text{otherwise} \end{cases} \quad (2.6)$$

The length of subsequences and the size of the dictionary for each dataset were chosen in order to obtain the best accuracy in the training set by using constrained DTW. In addition, the SimpleRank used to construct the dictionaries was also implemented by using the classic constrained DTW instead of the distance measure proposed in this work. This was done to ensure we are not biasing our experimental analysis in favor of our method.

Once created the dictionary, we need to estimate a good value for the parameter r . For this, we experimented with a wide range of possible values. We set r as a relative value to the length of the time series under comparison. Specifically, we used a set of values $rl_r \in$

$\{0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5\}$, such that $r = \lceil n \times rl_r \rceil$ and n is the length of the time series.

We limited the value of r to be at most half the number of observations of the time series in order to avoid meaningless alignments, such as the ones obtained by OBE-DTW in the example illustrated in Figure 12.

Besides defining a range of values to evaluate, we need to define a procedure to perform such evaluation. Note that the choice of the size of the dictionary is a crucial determinant of the time complexity of the algorithm. For this reason, the number of examples in the dictionary tends to be small in order to keep the algorithm fast, which makes learning r difficult if we use the data in the dictionary exclusively.

In order to learn the value of r , we used a validation set containing all the training time series but those chosen as part of the dictionary. However, we notice that cross-validation techniques on the training set lead to similar results.

2.6.2.1 Motor Current Data

Our first case study considers electric motor current signals. This dataset has long been a staple of researchers interested in prognostics and novelty detection (POVINELLI *et al.*, 2004). We refer the reader interested in the procedure to generate such data to (DEMERDASH; BANGURA, 1999).

The data in question includes 21 classes representing different operating conditions. In addition to a class that represents (a slight) diversity of healthy operation, the other classes represent different defects in the apparatus (in particular, one to ten broken bars and one to ten broken end-ring connectors).

The original data used in this study is segmented, but with no attention paid to avoiding endpoints inconsistencies. Therefore, in this case, we did not use the approach of simulating a data stream. We segmented the original time series using a static window placed in the middle of each time series. With this procedure, the signals have different endpoints in each different length we consider. Figure 15 shows the classification results.

Given that this dataset is a very clear case of badly-defined endpoints, these results show the robustness of our proposal. Over all lengths we experimented with, ψ -DTW beats DTW by a large margin. Specifically, ψ -DTW can achieve accuracy rates as high as 40% while the best result achieved by the classic DTW is lower than 12%.

2.6.2.2 Robot Surface and Activity Recognition

In this case study, we consider the classification of signals collected by the accelerometer embedded in a Sony ERS-210 Aibo Robot (VAIL; VELOSO, 2004). This robot is a dog-like model equipped with a tri-axial accelerometer to record its movements.

Figure 15 – Classification results obtained by varying the time series length on the Motor Current dataset

Source: Elaborated by the author.

Using the streaming data sets collected by this robot, we evaluated the classification accuracy in two different scenarios: surface and activity recognition. In the former scenario, the goal is to identify the type of surface in which the robot is walking on. Specifically, the target classes for this problem are carpet, field, and cement. Figure 16 shows the results for this dataset.

Figure 16 – Classification results obtained by varying the time series length on the Sony AIBO Robot Surface dataset

Source: Elaborated by the author.

In the second scenario, the aim is the identification of the activity performed by the robot. In this case, the target classes are the robot playing soccer, standing in a stationary position, trying to walk with one leg hooked, and walking straight into a fixed wall. Figure 17 shows the results obtained in this scenario.

In both scenarios evaluated in this study, the results obtained by ψ -DTW are generally better than the classic DTW. However, there is an important caveat to discuss. Despite the improvements in accuracy in most time series lengths, the accuracy obtained by ψ -DTW was the same or slightly worse than the performance of the classic DTW in a few experiments. This happened because our procedure to learn the relaxation factor was not able to find a more suitable value in these cases. Even in these cases, the poor choice of r did not significantly affect the classification accuracy.

Figure 17 – Classification results obtained by varying the time series length on the Sony AIBO Robot Activity dataset

Source: Elaborated by the author.

2.6.2.3 Gesture Recognition

Gesture recognition is one of the most studied tasks in the time series classification literature. The automatic identification of human gestures has become an increasingly popular mode of human-computer interaction.

In this study, we used the Palm Graffiti Digits dataset (ALON *et al.*, 2009), which consists of recordings of different subjects “drawing” digits in the air while facing a 3D camera. The goal of this task is the classification of the digits drawn by the subjects. Figure 18 shows the results.

Figure 18 – Classification results obtained by varying the time series length on the Palm Graffiti Digits dataset

Source: Elaborated by the author.

Similar to our findings with the robot data, the accuracy rates obtained by our proposal are usually better than the obtained by the classic DTW. In few cases, the accuracy is slightly worse. However, most important is the robustness of ψ -DTW to the cases where the prefixes and suffixes seem to significantly affect the classification. For instance, there is an expressive loss of accuracy obtained by the classic DTW in the dataset containing time series with 150 observations. The lost is notably less drastic when we using ψ -DTW.

2.6.2.4 Sign Language Recognition

Another specific scenario with gesture data used in this work is the recognition of sign language. A sign language is an alternative way to communicate by gestures and body language that replace (or augment) the acoustic communication. In this work we used a dataset of Australian Sign Language (AUSLAN) (KADOUS *et al.*, 2002). The original dataset is composed of signs separately recorded in different sections. We used 10 arbitrarily chosen signs of each recording session displaced as a data stream. Figure 19 shows the results.

Figure 19 – Classification results obtained by varying the time series length on the AUSLAN dataset

Source: Elaborated by the author.

In contrast to the previous gesture recognition case, the accuracy rates obtained by relaxing the endpoint constraint are always better for this dataset. More importantly, the best accuracy rates were significantly superior when using ψ -DTW.

2.6.2.5 Human Activity Recognition

Due to the growth in the use of mobile devices containing movement sensors (such as accelerometers and gyroscopes), there is also a notable increase in the interest of human activity analyses using this kind of equipment.

In this final case study, we investigate the robustness of ψ -DTW in the recognition of human activities using smartening accelerometers. For this purpose, we used the dataset that first appeared in (ANGUITA *et al.*, 2012). Originally, the recordings are composed of 128 observations of three coordinates of the device's accelerometers. In our study, we used the x -coordinate disposed in a streaming fashion. Figure 20 shows the results.

Again, the accuracy obtained by ψ -DTW is better than the obtained by the classic DTW in all the cases for this dataset. This success of these results is due to, in part, to a good choice of value to the relaxation factor. This is the main topic of the following section.

2.6.2.6 Summary of the Results

The results presented so far show that ψ -DTW achieves better results than the classical DTW in most of the experimented cases. In the case that such results are not clearly strong, a

Figure 20 – Classification results obtained by varying the time series length on the Human Activity Recognition dataset

Source: Elaborated by the author.

hypothesis test on the accuracies obtained by both methods may be used. For this, we performed a paired Wilcoxon signed-rank test for comparing the performances of DTW and ψ -DTW. Notice that we compared the algorithms using warping window in a separated test. Using a confidence factor of 95%, the test rejected the null hypothesis (that the medians of the accuracies are similar) for both cases, i.e., with or without the use of constrained-warping.

The results analyzed so far regard to each pair of dataset and time series length. However, it is interesting to analyze the accuracies obtained for each dataset, considering the best time series length. To evaluate this, we used the validation procedure applied to learn r as the method to choose the time series length to assess the performance of ψ -DTW. For comparison, we used the best accuracy obtained by OBE-DTW and DTW. Note that this analysis is favoring the competing algorithms, given that we used an oracle instead of learning the best series length for these methods. Table 1 shows the result of this experiment.

Table 1 – Accuracy obtained by OBE-DTW, DTW, and ψ -DTW

Dataset	OBE-DTW	DTW	ψ -DTW	cDTW	ψ -cDTW
AUSLAN	0.503	0.500	0.579	0.490	0.514
Human Activity	0.555	0.558	0.575	0.566	0.578
Motor Current	0.114	0.119	0.400	0.119	0.405
Palm Graffiti	0.262	0.374	0.355	0.363	0.363
Robot Activity	0.839	0.845	0.854	0.822	0.833
Robot Surface	0.950	0.846	0.910	0.842	0.842

Source: Research data.

2.7 Lower Bounding ψ -DTW

One of the biggest concerns while designing a new distance measure is time efficiency. This is more prevalent in our case since we are proposing a modification of Dynamic Time Warping, an $\mathcal{O}(n^2)$ algorithm. In fact, a straightforward implementation of the nearest neighbor

algorithm under DTW makes its use impractical on large datasets. For this reason, the community has proposed several methods to improve the efficiency of the similarity search under DTW.

Specifically, Rakthanmanon et al. (2012) (RAKTHANMANON *et al.*, 2012) shows that the combination of few simple techniques for speeding-up similarity search makes possible to handle truly massive data under DTW. We claim that all these methods can be applied to the ψ -DTW with subtle or no modifications.

Some of the most important speed up methods rely on the use of a lower bound (LB) function. A LB function returns a value certainly lower or equal to the true DTW between two objects. Our algorithm is amenable to adaptation of LB functions.

Before explaining how to adapt LB functions to ψ -DTW, we briefly explain the intuition behind the use of LB on time series similarity search. Consider that we have a variable best-so-far that stores the distance to the nearest neighbor known up to the current iteration of the search algorithm. We can use this information to decide if we can avoid the expensive calculation of DTW. In order to do this, for each time series in the training set, we first calculate the LB of the distance between it and the query. Clearly, if the LB function returns a value greater than the best-so-far, the training object is not the nearest neighbor of the query. Therefore, the current object can be discarded before having its distance to the query estimated. We can extend this to a k-nearest neighbor scenario by simply replacing the best-so-far by the distance to the k-th nearest object known at that moment.

Now we are in position to answer the following question. How can we use previously proposed LB functions with ψ -DTW?

We first note that ψ -DTW actually lower bounds the DTW, as exemplified in Figure 21. From a practical standpoint, the alignment path that starts at the first pair of observations and finishes by matching the last one is a possible alignment found by ψ -DTW that correspond to the exact classic DTW. Any other alignment found is considered optimal only in the case in which it provides a smaller value than the one obtained by DTW. This situation occurs when our method disregards some pair of observations that contributes to the total cost of matching.

Figure 21 – The distance between all the pairs of fifty time series objects in the AUSLAN dataset sorted by their DTW distances. In this experiment, we used both warping constraint and relaxation factor as 10% of the length of time series

Source: Elaborated by the author.

For this reason, it is not possible to apply most of the known LB functions directly to our method. Adapting a LB function to ψ -DTW requires the analysis of the possible first and last pairs of observations. For sake of exemplification, we will adopt the most widely used LB function, the LB_Keogh (KEOGH; RATANAMAHATANA, 2005).

The calculation of LB_Keogh consists of two main steps. The first step is the estimation of an envelope to a given query time series q of length n . Specifically, the envelope is composed of an upper sequence $U = (U_1, U_2, \dots, U_n)$ and a lower sequence $L = (L_1, L_2, \dots, L_n)$ defined by Equation 2.7.

$$\begin{aligned} U_i &= \max_{i-w \leq j \leq i+w} (q_j), 1 \leq i \leq n \\ L_i &= \min_{i-w \leq j \leq i+w} (q_j), 1 \leq i \leq n \end{aligned} \quad (2.7)$$

where w is the length of the warping constraint window. Clearly, the partials $i - w$ and $i + w$ are restricted to the extent of the query. Figure 22 exemplifies the upper and lower sequences of a given query time series.

Figure 22 – Upper and lower sequences of a given query time series q estimated by LB_Keogh

Once the envelope is calculated, we are in the position to estimate the value of the LB function. For each time series t to be compared to the query q , the value LB_Keogh is calculated as the Euclidean distance between the observations of t that falls outside the envelope and the nearest upper or lower sequence. Figure 23 illustrates this step in the comparison of the previously used query q and a specific time series t .

Figure 23 – The LB_Keogh is calculated by using the values of the time series t that fall outside the region bounded by the envelope

The only issue in directly applying LB_Keogh to lower bound ψ -DTW is the fact that it is constrained by the classic endpoint constraint of DTW. Therefore, in order to adapt LB_Keogh

to our method, we need to relax its endpoints. Since ψ -DTW can skip the matching of the first and last r observations in either q or t , the LB function should ignore these values. We call the adapted LB function ψ -LB_Keogh, and define it formally in Equation 2.8.

$$\psi - LB_{Keogh} = \sum_{i=r+1}^{n-r} \begin{cases} (t_i - U_i)^2 & \text{if } t_i > U_i \\ (L_i - t_i)^2 & \text{if } t_i < L_i \\ 0 & \text{otherwise} \end{cases} \quad (2.8)$$

Figure 24 illustrates the ψ -LB_Keogh between q and t .

Figure 24 – ψ -LB_Keogh ignores the values in the dashed regions

To be effective, a LB function has to present the following properties: (i) its calculation is fast; (ii) and it is tight, i.e., its value is close to the true DTW. Particularly, the pruning power of a lower bound function is directly related to its tightness.

To demonstrate the tightness of ψ -LB_Keogh, we compared it with the tightness of LB_Keogh for all the study cases in Section VI.B. We quantified the tightness of the LBs by dividing them by the corresponding DTW distances. In this experiment, we set the warping window as 10% of the time series length. The relaxation factor takes the same value. Table 2 shows the results obtained in the training set with the shortest time series used in each study case.

Table 2 – Tightness of LB_Keogh and ψ -LB_Keogh

Dataset	Tightness of LB_Keogh	Tightness of ψ -LB_Keogh
AUSLAN	0.522	0.484
Human Activity	0.173	0.152
Motor Current	0.259	0.292
Palm Graffiti Digits	0.549	0.490
Sony Robot Activity	0.120	0.110
Sony Robot Surface	0.174	0.151

Source: Research data.

From these results, we can note that the tightness of both methods is similar. In fact, ψ -LB_Keogh is even tighter than LB_Keogh in one of the experimented datasets. This indicates that endpoint constraint relaxation does not impair the tightness of ψ -LB_Keogh.

2.8 Conclusion

In this paper, we proposed a modification of the endpoint constraint of DTW to make it suffix- and prefix-invariant. In addition to be simple and intuitive, our method is quite effective. Experimental results show that our method outperforms the classic DTW by a large margin in various datasets that contain spurious endpoints. In addition, we demonstrated that the distance obtained by our method can be tightly lower bounded by a slight modification of the current lower bounds of DTW, which indicates that our modified DTW is tractable for large datasets.

For the sake of clarity and brevity in this work we only discussed the application of our algorithm to classification. However, it can also be applied to a large variety of tasks, such as clustering, motif discovery, outlier detection, etc. We leave those explorations for future work.

SPEEDING UP ALL-PAIRWISE DYNAMIC TIME WARPING MATRIX CALCULATION

Abstract: Dynamic Time Warping (DTW) is certainly the most relevant distance for time series analysis. However, its quadratic time complexity may hamper its use, mainly in the analysis of large time series data. All the recent advances in speeding up the exact DTW calculation are confined to similarity search. However, there is a significant number of important algorithms including clustering and classification that require the pairwise distance matrix for all time series objects. The only techniques available to deal with this issue are constraint bands and DTW approximations. In this paper, we propose the first exact approach for speeding up the all-pairwise DTW matrix calculation. Our method is exact and may be applied in conjunction with constraint bands. We demonstrate that our algorithm reduces the runtime in approximately 50% on average and up to one order of magnitude in some datasets.

3.1 Introduction

Dynamic Time Warping (DTW) is certainly the most relevant distance for time series analysis. Such relevance has been evidenced by a large body of experimental research showing that, for instance, the 1-nearest neighbor DTW (1-NN-DTW) algorithm frequently outperforms more sophisticated methods on a large set of benchmark datasets ([WANG *et al.*, 2013](#)).

The main issue with DTW is its computational complexity. A straightforward implementation of DTW is quadratic in time and space. Although a simple trick can make DTW linear in space¹, the time complexity is a more difficult matter.

An important observation is that all the recent advances in speed up DTW calculations are confined to similarity search. However, there is a significant number of data mining algorithms,

¹ When the only output of interest is the final distance and the warping path can be disregarded.

including clustering and classification, that require the pairwise distance matrix for all time series objects.

In the particular case of time series clustering, the authors of (ZHU *et al.*, 2012) show that the calculation of the all-pairwise distance matrix for DTW completely dominates the runtime of well-known clustering algorithms. For instance, for a large dataset, the computation of the all-pairwise distance matrix would take approximately 127 days using off-the-shelf desktop computers and a naïve quadratic DTW algorithm. On the same computer, the calculation of a hierarchical clustering given an already computed pairwise distance matrix would take only 4 seconds.

In case a researcher or practitioner is interested in applying DTW with algorithms that require the all-pairwise distance matrix, the only speed up techniques available are constraint bands (also known as warping windows) (SAKOE; CHIBA, 1978; ITAKURA, 1975) or DTW approximations (SALVADOR; CHAN, 2007; SPIEGEL; JAIN; ALBAYRAK, 2014).

In this paper, we propose a novel approach for speeding up the all-pairwise DTW matrix calculation. Our method uses an upper bound estimation to prune unpromising warping alignments. In other words, our method prunes DTW partial paths that will lead to unfruitful warping paths by comparing the current calculated value with a distance upper bound.

3.2 Background

Euclidean distance (ED) is the most established distance measure between time series. The ED measures the dissimilarity between time series comparing the observations at the exact same time. For this reason, the ED can be very sensitive to distortions in the time axis. Many applications require a more flexible observation matching, in which an observation of the time series x_i at time i can be associated to an observation of the time series y_j at time $j \neq i$.

The DTW distance achieves an optimal nonlinear alignment of the observations under boundary, monotonicity and continuity constraints. DTW is usually calculated using a dynamic programming algorithm. Equation 3.1 describes the initial condition of the algorithm².

$$dtw(i, j) = \begin{cases} \infty, & \text{if } i = 0 \text{ or } j = 0 \\ 0, & \text{if } i = j = 0 \end{cases} \quad (3.1)$$

² Further in this paper, we will assume that the time series objects may have different lengths. Therefore, $x = x_1, x_2, \dots, x_N$ and $y = y_1, y_2, \dots, y_M$.

Equation 3.2 presents the recurrence relation of DTW algorithm.

$$dtw(i, j) = c(x_i, y_j) + \min \begin{cases} dtw(i-1, j) \\ dtw(i, j-1) \\ dtw(i-1, j-1) \end{cases} \quad (3.2)$$

where $i = 1 \dots N$ and $j = 1 \dots M$. $c(x_i, y_j)$ is the cost of matching two observations x_i and y_j , usually calculated with squared Euclidean distance.

The resulting value in $dtw(N, M)$ is the DTW distance between x and y . Thus, the algorithm iteratively fills an array with the lowest accumulated cost for all alignments to each pair of observations to be matched. Figure 25 shows an example of the optimal non-linear alignment found by this algorithm and how it is represented in the DTW calculation matrix.

Figure 25 – Optimal non-linear alignment (*left*) and the matrix obtained by the dynamic programming algorithm, highlighting the optimal alignment (*right*)

Source: Elaborated by the author.

In order to improve the efficiency of DTW calculations, the use of warping windows is common (SAKOE; CHIBA, 1978; ITAKURA, 1975). Warping window, or constraint band, defines the maximum allowed time difference between two matched observations. From the algorithm standpoint, this technique restricts the values that need to be computed to a smaller area around the main diagonal of the matrix.

However, the exact window size that would provide the best results for a dataset is data dependent. Outside classification problems with 1-NN, there are no clear guidelines to set this parameter and possibly the best approach is to evaluate the results for several window sizes. We will return to this topic in the Section 3.6.2.

3.3 On the Need of the All-Pairwise Distance Matrix

Although the scientific community has mainly focused on the speed up of DTW calculation for similarity search, other algorithms, for instance in clustering and classification require the all-pairwise DTW matrix.

Recently, the community has devoted some effort to improve the efficiency of time series clustering algorithms. For instance, Zhu et al. have framed the problem of calculating the all-pairwise DTW matrix as an anytime algorithm and applied such approach to clustering (ZHU *et al.*, 2012) and Ulanova et al. have proposed new techniques to cluster temporal objects in admissible time (ULANOVA; BEGUM; KEOGH, 2015).

The need of the all-pairwise distance matrix is easily seen in the clustering task. Many clustering algorithms do not require the objects as input, but their relations. In the case of time series mining, the relations can be defined as the distance between each pair of instances. For example, this is the case of most the methods of the family of hierarchical clustering and the well-known k -medoids. Besides k -means be probably the most popular clustering method and not be relational, the average point of time series objects is not trivial in a DTW space. The best one can do is to make use an approximate average point as centroid (PETITJEAN; KETTERLIN; GANÇARSKI, 2011).

The need of all-pairwise distance matrix is not restricted to clustering algorithms. A recent advance in time series classification proposes the use of machine learning classifiers with distances representing values of attributes (KATE, 2016). Being n the number of training examples, this approach constructs an $n \times n$ attribute-value table in which the values of attributes are defined by the distance of one object to all other objects in the training set. Therefore, in order to construct a classification model, this approach needs the whole distance matrix between the training examples. For classifying a new example, it needs the distance between the query time series and all the training examples.

3.4 Related Work

Due to the relevance of DTW in time series mining and its high computational cost, the scientific community has proposed several approaches to deal with DTW in tasks involving large amounts of data.

3.4.1 Similarity Search

Although similarity search is not the focus of this work, we briefly review the intuition of these approaches to clarify the reasons they are not applicable to all-pairwise distance calculations.

Suppose that we are interested in finding the k most similar objects to a query time series and a variable *best-so-far* stores the true DTW distance to the k -th nearest object known up to a certain moment of the algorithm execution. Consider a lower bound function (LB) that returns a value that is certainly lower or equal the true DTW between two objects. Clearly, if the LB between a given training object (o_k) and a query time series (q) is higher than the *best-so-far*, we know that o_k is not one of the k nearest neighbors of q . Therefore, such an object can be discarded. It is obviously only applicable when we have no interest in those objects whose distance are greater than the *best-so-far*.

The main challenge on speeding up the *exact* computation of the all-pairwise matrix is that we cannot rely on techniques that avoid computing the DTW distance for certain pairs of objects. *The only resource is to improve the internal efficiency of the DTW computation.*

3.4.2 DTW Approximations

Distance functions that approximate DTW are a popular approach to speed up DTW calculations. Since these methods only require two time series objects as input, they are directly applicable to calculating the all-pairwise distance matrix.

Several approaches were proposed in the literature with this purpose. Some examples are FastDTW (SALVADOR; CHAN, 2007) and Lucky Time Warping (SPIEGEL; JAIN; AL-BAYRAK, 2014). Approximations may also be used to fill the distance matrix in anytime fashion (ZHU *et al.*, 2012).

The main drawback with these approaches is that they do not provide any guarantees in terms of approximation error to the true DTW. In other words, the user has no means of setting an allowable maximum error in reference to the true DTW.

3.4.3 Biological Sequences Alignment

The problem of time series alignment is similar to the alignment of biological sequences, such as proteins and RNA. Usually, biological sequences are also compared with costly similarity functions.

In order to improve the running time of sequence alignment algorithms, Carrillo and Lipman (CARRILLO; LIPMAN, 1988) proposed to expand the function calculations only to partial alignments that present promising values. First, the approach calculates a lower bound to the similarity function and use this value as the threshold to decide which cells should be expanded in the DTW matrix. Given a change in any partial alignment, the cells affected by the changed value are calculated only if the current value is higher than the threshold defined by the lower bounding function. Although simple, this procedure guarantees the calculation of the exact similarity value.

This strategy was never used directly to speed up DTW calculations. However, a similar idea is used as an intermediate step of a similarity search algorithm named FTW, described next.

3.4.4 FTW

Fast search method for dynamic Time Warping (FTW) (SAKURAI; YOSHIKAWA; FALOUTSOS, 2005) is the method that shares more similarities with our proposal. The authors propose a similarity search method based on a recursive refinement of a lower bound calculation.

In a coarse representation, the method finds a lower bound to DTW, using a dynamic programming algorithm. If the computed lower bound is smaller than the *best-so-far*, the method proceeds to a finer representation. Otherwise, the calculation is aborted, since the object is not one of the nearest neighbors.

In each level of approximation, the method prunes the DTW matrix values that are guaranteed to be greater than the *best-so-far*. Such an algorithm, called EarlyStopping, is the most similar method to our proposal. However, our method does not rely on a *best-so-far*, which would restrict it to similarity search. Instead, we use an upper bound, which can be initially set as ED and refined as the algorithm proceeds in the calculation of the DTW distance.

In summary, although FTW and our proposal share some similarities, FTW is a similarity search algorithm and relies on a *best-so-far*. So, it is not directly applicable to calculate the all-pairwise DTW matrix. Our method prunes warping paths that are greater than an upper bound in the actual DTW calculation and is not dependent of a *best-so-far*. Finally, FTW increasingly refines time series representations while our method is computed in the full resolution only.

3.5 DTW with Pruned Warping Paths

In this paper, we propose the DTW with Pruned Warping Paths (PrunedDTW). We adapted the traditional DTW algorithm to recognize and prune cells in the DTW matrix that are guaranteed to not lead to alignments that will result in the optimal path.

We emphasize that our method is similar or considerably faster than DTW and always returns the optimal path between two time series objects. In addition, PrunedDTW can be implemented in linear space. The additional variables necessary for pruning are $\mathcal{O}(1)$ and for updating the upper bound is $\mathcal{O}(\max(N, M))$ in space. Finally, PrunedDTW supports warping windows and time series of different lengths.

Even more importantly, PrunedDTW is orthogonal to all of the proposals of speeding DTW up that we are aware of. Therefore, PrunedDTW can be used in conjunction with the literature to further increase the efficiency of DTW in different tasks, including similarity search.

3.5.1 The Intuition Behind our Proposal

Our method is motivated by a very simple observation: frequently, the cells of a DTW matrix vary in a large range of values. Usually, the values around the optimal alignment in the matrix are relatively low. In contrast, even cells moderately distant from the optimal path frequently receive much higher values. Figure 26 shows an example of DTW matrix for two time series from the Mallat dataset. There are several regions in the matrix with accumulated costs that go far beyond the optimal DTW distance (which is 71.77 in this case).

Figure 26 – DTW matrix between two time series. The colors indicate the value obtained in each cell of the matrix

Source: Elaborated by the author.

A cell at a position (i, j) of the DTW matrix represents the cost of the optimal alignment that starts at the position $(1, 1)$ and ends at (i, j) . For an internal position in the DTW matrix, i.e., $i < N, j < M$, if the cell at (i, j) is part of the optimal path then the optimal path cost has the cost at (i, j) plus the cost of matching observations from (i, j) up to (N, M) . As the cost of matching two observations is zero or positive value, the warping paths are monotonically increasing.

Intuitively, if a cell at (i, j) has a high value, it clearly indicates that such a warping path is very unlikely to lead to the optimal path. However, as we want to propose an exact method, we need to establish a threshold in which we can guarantee that such a partial warping path will not be part of the optimal path. More importantly, we need a *pruning strategy* that explores the cells with large values and the DTW recurrence relation of Equation 3.2 to decide when we can start and stop evaluating the cells. We detail the pruning strategy and the corresponding algorithm in the next section.

3.5.2 Pruning Strategy

In order to define which values are high enough to be pruned, we can use an upper bound (UB) of the DTW distance. In this paper, we use the squared Euclidean distance (sqED) as UB³. This distance measure is a special case of DTW when the optimal alignment corresponds to the main diagonal of the DTW matrix.

³ Our approach also works for time series of different lengths in which the sqED is not defined. We will return to this topic in Section 3.5.3.

Our algorithm calculates the UB as the first step. We note, however, that it represents a small overhead since the UB calculation is linear in time complexity. After estimating the UB, we must specify a criterion so that we can prune the calculation of DTW matrix cells, ensuring not to discard any element that may belong to the optimal alignment. For this purpose, we use different criteria for pruning the beginning and the end of each row⁴ of the matrix. Algorithm 4 details the proposed method.

Algorithm 4 – PrunedDTW algorithm

Requer: Time series x , with length N

- 1: Time series y , with length M
- 2: Warping window size ws
- 3: Upper bound UB of the DTW between x and y

Assesure: The distance between x and y according DTW

```

4:  $sc \leftarrow 1$                                 ▷ Auxiliary variable to prune lower triangular
5:  $ec \leftarrow 1$                                 ▷ Auxiliary variable to prune upper triangular
6: para  $i \leftarrow 1$  até  $N$  faça                ▷ Initialize the matrix of DTW calculations
7:    $D[i, 0] \leftarrow \infty$ 
8: fim para
9: para  $i \leftarrow 1$  até  $M$  faça
10:    $D[0, i] \leftarrow \infty$ 
11: fim para
12:  $D[0, 0] \leftarrow 0$ 
13: para  $i \leftarrow 1$  até  $N$  faça
14:    $beg \leftarrow \max(sc, i - ws)$ 
15:    $end \leftarrow \min(i + ws, M)$ 
16:    $smaller\_found \leftarrow FALSE$ 
17:    $ec\_next \leftarrow i$ 
18:   para  $j \leftarrow beg$  até  $end$  faça
19:      $D[i, j] = sqED(x_i, y_j) + \min(D[i - 1, j - 1], D[i - 1, j], D[i, j - 1])$ 
20:     Pruning strategy                                ▷ copy & paste Algorithm 5 here
21:   fim para
22:    $ec \leftarrow ec\_next$ 
23: fim para
   retorna  $D[N, M]$ 

```

Line 20 represents the pruning technique (not presented so far) that modifies the regular DTW algorithm. The pruning technique manipulates the values of the auxiliary variables to determine the beginning of the calculation of each row (line 14 of the algorithm) and its end (which depends on the assignments made in the lines 17 and 22). Algorithm 5 describes the pruning criteria in details.

The main idea of the pruning strategy is that the values related to a row i define which column can be pruned in the row $i + 1$. In particular, the variables sc (start column) and ec (end column) control the range of columns that needs to be analyzed in the next row.

⁴ Our implementation traverses the matrix in a row-major order. However, the algorithm can also be implemented by traversing the matrix in a column-major order.

Algorithm 5 – Pruning criteria implementation

```

1: se  $D[i, j] > UB$  então
2:   se  $smaller\_found = FALSE$  então
3:      $sc \leftarrow j + 1$ 
4:   fim se
5:   se  $j \geq ec$  então
6:     break ▷ break the for loop / jump to the next row
7:   fim se
8: senão
9:    $smaller\_found \leftarrow TRUE$ 
10:   $ec\_next \leftarrow j + 1$ 
11: fim se

```

There are two separate strategies, one for pruning cells in the lower triangular matrix and the other one for the upper triangular. For the lower triangular, the pruning is controlled by the variable sc . The idea is that, traversing a row from left to right, as long as we find columns with values greater than UB, it is safe to say that the same columns on the next row will also have values greater than UB.

Figure 27 illustrates this idea. For the row 4, the first two columns have a value greater than UB. Therefore, the variable sc is set to column 3 (Algorithm 5, line 3) and the processing can safely start at column 3 in the next row. We can prune the computation of the variables A and B because of the DTW recurrence relation represented by the three arrows. The value of the cell at (i, j) is the cost of matching the observations x_i and y_j added to the minimum of the values in three other cells of the matrix (Algorithm 4, line 19). As the column 0 is initialized with infinity values, the variable A will obligatorily have a value greater than UB. The same occurs to B which depends on $A > UB$ and other two cells in the previous row also greater than UB. In contrast, variable C may have a value smaller than UB since it depends on $D(4, 3) \leq UB$.

Figure 27 – Pruning in the lower triangular matrix

Source – Elaborated by Dr. Gustavo E. A. P. A. Batista

The initial value of the variable sc is 1, i.e., while no values greater than UB are found, the calculation in each row will start at the first column. In the case that a warping window is used, the rows will be initiated in the column with the highest index between the column established by the warping window and the one related to the pruning criteria (Algorithm 4, line 14).

Notice that the main diagonal cells (in red) are marked as smaller than or equal to UB. This occurs because, as mentioned before, all warping paths are monotonically increasing and the main diagonal corresponds to ED. This ensures that for a row i , $sc \leq i$ and our pruning is confined to the lower triangular matrix. This is an important observation, given that we will later update the UB as we fill the DTW matrix.

The second pruning strategy is responsible for pruning the upper triangular matrix. This strategy defines the column where we can stop the calculation of the next row. The variable ec stores the column where the first of a sequence of values greater than UB starts and goes until the end of the current row.

Figure 28 illustrates the idea. In this example, row 1 is processed and ec is set to 4. The variable ec is marking the first value of a sequence of values greater than UB that finishes at the end of the row (Algorithm 5, line 10). We can stop the row 2 as soon as two criteria are met: (i) the calculated value is greater than UB and (ii) the current column index is greater or equal to ec . Suppose that the cell A is greater than UB. In this case, criterion (ii) is not met. We can see that cell B may be smaller than UB since it can use $D(1, 3)$ which is lower or equal to UB. However, if B is greater than UB then both criteria are met and we can stop processing row 2. This occurs because variables C and D can only inherit values from the matrix that are greater than UB.

Figure 28 – Pruning in the upper triangular matrix

	0	1	2	3	4	5	6
0	0	∞	∞	∞	∞	∞	∞
1	∞	$\leq UB$	$\leq UB$	$\leq UB$	$> UB$	$> UB$	$> UB$
2	∞		$\leq UB$	A	B	C	D
3	∞			$\leq UB$			

ec

Source – Elaborated by Dr. Gustavo E. A. P. A. Batista

There is a subtle detail in the algorithm that allow us to initialize $ec = 1$, instead of $ec = M$. This occurs because of the initialization of the DTW matrix with infinite values. Therefore, $ec = 1$ actually marks the first column in row 0 that is greater than UB.

3.5.3 Iteratively Updating the Upper Bound

An interesting point about the DTW algorithm is that the matrix D stores the costs of the optimal paths from $(1, 1)$ to (i, j) . This means we can use such partial optimal matching to update the UB value. In the case where sqED is adopted as UB, every time we compute a main diagonal cell $D(i, i)$ we are in position to update UB.

For this purpose, we also need the partial values of the UB calculation. Since the calculation of sqED does not depend on the order in which we calculate the distance between each pair of observations, we calculate this measure in reverse order, i.e., from the end to the

beginning of the time series. At each step of the calculation, we store the partial value obtained in a vector $sqED_{partials}$, according to Equation 3.3. Notice that the calculation of $sqED_{partials}$ is still $O(N)$ when computed in the reverse order.

$$sqED_{partials}(i) = \sum_{j=i}^N (x_j - y_j)^2 \quad (3.3)$$

Once we computed $D(i, i)$, we have all the necessary information to update the UB, according to Equation 3.4.

$$UB = D(i, i) + sqED_{partials}(i + 1) \quad (3.4)$$

This equation updates the UB with the optimal DTW alignment of the first i observations summed to the squared Euclidean distance between the observations ahead of i . Thus, the value of UB becomes increasingly tighter in relation to the optimal DTW value, as we proceed in the computation of the matrix. Notice that $D(i, i)$ is always smaller or equal to the Euclidean distance of the first i observations of the time series. Therefore, the pruning power of our method increases at each iteration.

At first glance, the use of Euclidean distance as UB restricts our proposal to the comparison of time series of the same length. Such restriction occurs because ED is only defined for objects of the same length. However, we can easily circumvent such restriction. Let N be the length of the shorter time series. A UB can be obtained by calculating the squared Euclidean distance between the first N observations summed to the distance between the remaining values of the longer time series to the last observation of the shorter time series. There are other possible UB since any warping path is a UB for DTW. We will further explore this fact in the next section.

Our implementation has some additional features such as a linear space complexity. We do not describe such features in details here because they are not the main contributions of this paper. We have built a website in which we made available all detailed numerical results, source code and supplemental material not included in this paper (SILVA; BATISTA, 2016a). However, we note that our paper is completely self-contained. In addition, our website provides a proof of the correctness of our algorithm⁵.

3.5.4 Other UB Approaches

So far we have only considered the sqED as UB. However, any measure that is an upper bound of DTW can be used, as far as the cost of its calculation does not compromise the overall cost of the algorithm.

⁵ In this thesis, this proof can be seen in Section 4.4

The use of other UB approaches has two immediate consequences for our work. The first one is that we can use the true DTW distance as UB. Even if such an approach is not practical in real situations, it provides us the best-case analysis in which the algorithm would prune the highest possible number of cells. Figure 29 shows an example of this fact. Note how the pruning in is much more aggressive by using the actual DTW as UB. The use of a tight UB may even prune cells in the main diagonal of the matrix.

Figure 29 – Regions of the DTW matrix pruned by our proposed criteria (in white) by using the sqED (left) and the true DTW (right) upper bounds for the same pair of time series. The red lines show the positions of the main diagonal cells

Source: Elaborated by the author.

The second consequence will be better explored in Section 3.6.4. In most of experimental analyses, we need to perform a search for the best warping window size. The naïve approach for this is simply run the DTW algorithm multiple times, calculating the all-pairwise matrix for each warping window size. However, we can speed up this approach by using the optimal DTW distance calculated for a smaller warping window as UB for the next (larger) window size to be assessed.

3.6 Experimental Evaluation

As we are proposing an exact method, the main way of assessing our results is the running time of the algorithm for the computation of the all-pairwise distance matrix.

We were *extremely* careful to measure the runtimes of the algorithms in order to provide meaningful experimental results. We used *identical* DTW implementations with and without the pruning method. Therefore, the difference in time between them can be only explained by the proposal in this paper.

We ran all the experiments on the same computer⁶. At any time, there was only one process computing DTW distances. In order to reduce the variance caused by other processes outside our control⁷, we executed each method three times and reported the average running times.

We notice that we are very committed to the reproducibility of our results. For this purpose, the web page for this paper (SILVA; BATISTA, 2016a) contains all source code and detailed results obtained in our experiments.

3.6.1 Benchmark Datasets

In order to assess the efficiency of PrunedDTW, we performed an experimental evaluation using 10 freely-available benchmark datasets. Specifically, all datasets are from the UCR Time Series Classification/Clustering Page (CHEN *et al.*,).

We chose data sets with time series with at least 500 observations. Although our method can be used with time series of any length, the calculation of all-pairwise distance matrix of short time series can be done relatively fast with the traditional algorithm.

3.6.2 On the Warping Window Length

Our experimental analysis is highly dependent on the warping window length. For this reason, before we introduce our results, we will discuss the relevance of this parameter. Empirical evidence points to the fact that small window sizes provide superior 1-NN classification accuracy (RATANAMAHATANA; KEOGH, 2005). On the other hand, there are no conclusive studies about this parameter for different algorithms or mining tasks, including clustering.

The assumption that small windows are the most suitable for time series matching is commonly accepted in the literature. However, there are few exceptions. An example is the classification method proposed by (KATE, 2016). In their empirical evaluation, the addition of DTW features calculated with no warping window improved the results considerably in comparison to the classifier that just uses features with constrained DTW. Actually, such improvement allowed their method to obtain a statistically significant difference over 1-NN-DTW.

We performed a quick experiment to evaluate whether the assumption of small warping windows also applies to some well-known clustering algorithms. We evaluated the performance of the hierarchical clustering with complete linkage and k -medoids algorithm using eight different values for warping window length: 5%, 10%, 15%, 20%, 30%, 40%, 50%, and 100% of the time series length. For each of the 10 datasets evaluated in our experiment, we measured the rand index and the silhouette.

⁶ The experiments were carried out in a desktop computer with 12 Intel(R) Core(TM) i7 – 3930K CPU @ 3.20GHz and 64Gb of memory running Debian GNU/Linux 7.3.

⁷ Such as process running operating system tasks.

Figure 30 summarizes the results. This graph presents the count of cases in which the best result for a given clustering algorithm and an evaluation measure was obtained for each value of warping window length. In case of a tie, all tied windows are counted. The distribution of the best results among the different lengths supports the recommendation to researchers and practitioners to look for several values of window length for new data sets, including the DTW with no warping window.

Figure 30 – Counting of cases in which the best result was obtained by each evaluated warping window length

Source: Elaborated by the author.

3.6.3 Runtime for All-Pairwise DTW Matrices

Given the importance of warping windows in both accuracy and execution times, we performed our experiments varying such parameter. We evaluated the time to calculate the distances matrices using windows with relative length of 10%, 20%, 30%, 40%, 50%.

Figure 31 graphically shows the results. We present the results of the three methods: DTW stands for the standard DTW algorithm, PrunedDTW is our proposal using sqED as UB and OracleDTW is our proposal with the true DTW as UB. Although OracleDTW cannot be used in practice, its results are optimal in the sense that they represent the highest possible number of pruned cells that our method could achieve with a (imaginary) perfect constant time UB. So, OracleDTW provides a reference of the best performance that could be obtained by PrunedDTW.

PrunedDTW outperformed DTW in most of the cases. This indicates that the overhead of the pruning tests (which increases the constant of the computational complexity) are usually compensated by the number of pruned cells. In some cases in which the warping window is small (usually 10%), PrunedDTW could not achieve a significant speed up. This is expected since warping windows significantly reduce the number of cells that need to be computed, leaving little space for improvements. As the warping windows are increased, PrunedDTW obtains more significant speedup. For a 50%-band size, the speedup of PrunedDTW relative to the computation time of DTW are in the range of 12.01% to 89.61%.

Figure 31 – Time (in seconds) to calculate the all-pairwise DTW distances with different warping window sizes

Source: Elaborated by the author.

The gain obtained by PrunedDTW over DTW is commonly larger than the gain of OracleDTW over PrunedDTW, indicating that ED was able to obtain most of the achievable speedup for these datasets. However, the cases in which it is not true to reveal the need of research for other upper-bound methods.

We note that we have performed additional experiments with DTW approximations as UB, whose results are not described in this paper. The ED presented the best compromise between tightness to DTW and time complexity among the evaluated functions.

3.6.4 Accumulative Runtime for All-Pairwise DTW Matrices

The results in Figure 31 represent the scenario in which the user knows which warping window size is the best for its application domain. A more realistic experiment would involve computing the running times for the calculation of pairwise distance matrices for several warping window sizes, simulating a search for an optimal value to this parameter.

As we have shown in the Section 3.6.2, the search for the optimal value of warping window length is required for a good performance of some time series mining algorithms. For this reason, we believe that it is important for any DTW speed up method to support warping windows so that the user can observe the impact of such parameter over the data at hand.

In the case of the interest in calculating the all-pairwise DTW matrix with the conventional algorithm, the trivial approach is to run the algorithm multiple times. In that case, no information is reused. This occurs because the newly computed cells are very likely to influence the already computed cells. In that case, most of the matrix cells would have to be recomputed.

Regarding PrunedDTW, we can simply use the distance computed for a smaller window size as UB for a larger window size. A distance computed for a smaller warping window is always smaller or equal to the sqED and, therefore, can be a more effective UB than sqED. For the first run, i.e., for the smallest window, there is no previous DTW distance to be used as UB. In this case, we can naturally adopt the sqED as UB.

Figure 32 presents the results. There is a significant difference between these results and the ones presented in Figure 31. In Figure 32 the running times for a given warping window size r means the *accumulative* time necessary to calculate the matrices for all warping windows smaller and equal to r . In this setting, PrunedDTW is even more effective, obtaining very similar speedups to the ones obtained with OracleDTW.

3.6.5 Comparison with FastDTW

As we are not aware of other approaches to speed up the exact all-pairwise DTW computation, it is natural to find difficulties to compare to the state-of-the-art. One possibility is the comparison to approximate DTW methods. This is not an entirely fair comparison, since the approximate methods will trade approximation accuracy by speed when PrunedDTW has zero approximation error.

We chose to compare our approach to FastDTW, since it is the most well-known approximation of DTW in the literature. However, we believe that our method is superior to FastDTW, for the following reasons: our method is exact, FastDTW is approximated; our method supports warping windows, FastFTW only works with unconstrained matrices; our method has no parameters, FastDTW has a parameter r (radius) that influences its accuracy and running time; and

Figure 32 – Accumulative time (in seconds) to calculate the all-pairwise DTW distances with different warping window sizes. The distance calculated for a smaller window size is used as UB for the next larger window size

Source: Elaborated by the author.

our method is simple to understand and implement, FastDTW is a sophisticated method with intricate details such as increasing data resolutions.

Since FastDTW does not support Sakoe-Chiba band, we evaluated the time to calculate the all-pairwise matrix for the unconstrained DTW. Table 3 presents the results in percentage of the time spent by conventional DTW algorithm.

Both methods are more efficient than the conventional DTW for all datasets. Although we noticed it is an unfair comparison, PrunedDTW achieved better or similar results (differences under 7%) than FastDTW in half of the evaluated datasets. Besides the fact that FastDTW

Table 3 – Percentual runtime of FastDTW and PrunedDTW regarding the conventional DTW algorithm

Dataset	FastDTW	PrunedDTW
Car	40.53%	39.68%
CinC ECG Torso	26.52%	80.02%
Haptics	28.01%	63.99%
InlineSkate	23.29%	63.96%
Lightning-2	43.85%	79.05%
MALLAT	27.63%	31.74%
Non-Invasive Fetal ECG Thorax1	37.53%	32.98%
Non-Invasive Fetal ECG Thorax2	37.26%	33.71%
Olive Oil	38.21%	10.96%
Starlight Curves	30.52%	61.95%

Source: Research data.

achieved better runtime results in the other 5 datasets, they are the datasets in which results of PrunedDTW are far from OracleDTW, reinforcing the need of new upper bound approaches.

3.7 Conclusion and Future Work

In this paper, we proposed a novel method to speed up the calculation of Dynamic Time Warping between time series. Differently from the previous work in literature, our method does not rely on a *best-so-far* or distance approximations. So, our proposal is adequate for any application that requires the distance between every pair of objects. The results show that our method is faster than the conventional DTW algorithm, especially in cases in which we want to calculate distances for different warping windows values or we want to use large constraint bands or unconstrained DTW.

As for the future work, we intend to explore the application of our proposal in different time series mining tasks and also in multidimensional time series scenarios. We also intend to evaluate the impact of different upper bounds. A possible direction to find new upper bound functions is to look for adaptations of bounding functions used in other applications domains, such as the computational biology.

SPEEDING UP SIMILARITY SEARCH UNDER DYNAMIC TIME WARPING BY PRUNING UNPROMISING ALIGNMENTS

Abstract: Similarity search is the core procedure for several time series mining tasks. While different distance measures can be used for this purpose, there is clear evidence that the Dynamic Time Warping (DTW) is the most suitable distance function for a wide range of application domains. Despite its quadratic complexity, research efforts have proposed a significant number of pruning methods to speed up the similarity search under DTW. However, the search may still take a considerable amount of time depending on the parameters of the search, such as the length of the query and the warping window width. The main reason is that the current techniques for speeding up the similarity search focus on avoiding the costly distance calculation between as many pairs of time series as possible. Nevertheless, the few pairs of subsequences that were not discarded by the pruning techniques can represent a significant part of the entire search time. In this work, we adapt a recently proposed algorithm to improve the internal efficiency of the DTW calculation. Our method can speed up the UCR suite, considered the current fastest tool for similarity search under DTW. More important, the longer the time needed for the search, the higher the speedup ratio achieved by our method. We demonstrate that our method performs similarly to UCR suite for small queries and narrow warping constraints. However, it performs up to five times faster for long queries and large warping windows.

4.1 Introduction

Following the remarkable availability of temporal data, time series mining is becoming a necessary procedure in a wide range of application domains. The estimate of a distance or similarity value between time series objects or subsequences is a common subroutine for several temporal data mining tasks. Consequently, the choice of the distance measure adopted to

compare the time series may harshly affect the performance of most distance-based algorithms. The scientific community has shown that the Dynamic Time Warping (DTW) is arguably the most suitable distance measure for a wide range of applications and mining tasks, such as classification (WANG *et al.*, 2013; KATE, 2016), clustering (BEGUM *et al.*, 2015), and pattern matching (CHAVOSHI; HAMOONI; MUEEN, 2016).

The similarity search consists of finding the most similar subsequence of a given query in a long reference data. For some applications, it may be extended to the k -nearest neighbor search, i.e., when the user is interested in finding a group of k similar subsequences.

A straightforward implementation of DTW is quadratic regarding time and space complexities. With the speed and the amount of data collected in several applications, this makes the search under DTW impractical. However, Rakthanmanon *et al.* (2012) have introduced the UCR suite, a set of optimizations that make the subsequence similarity search under DTW even faster than Euclidean distance with the techniques considered state-of-the-art up to that moment. Specifically, that work mainly consists of lower-bounding and early-abandon methods to discard nearest neighbors candidates before the computation of DTW. In most cases, the UCR suite can avoid the need for a DTW distance calculation.

Regarding the problem of finding the best match of a small subsequence in a long time series, Rakthanmanon *et al.* (2012) claim that “*for the problem of exact similarity search with arbitrary length queries, our UCR suite is close to optimal*”. In fact, the authors use a large set of experiments to support this claim. However, while the UCR suite approaches the optimality in avoiding the DTW calculation, such costly operation is still required for a relatively small percentage of the time series. Even being performed only to a small fraction of the subsequences, the DTW computation still represents a significant amount of the similarity search runtime.

A simple experiment can illustrate this fact. When searching a query in an electrocardiography (ECG) dataset with approximately 30 million data points, the DTW is calculated for only 4% of the total number of assessed subsequences. It demonstrates the extraordinary ability of the pruning techniques in avoiding DTW calculations. Even with this notable reduction of computations, the time for estimating the distance between the query and the assessed subsequences corresponds to approximately 60% of the entire search runtime. This cost is even higher in some cases, depending on the parameters of the similarity search, such as the query length.

In this work, we propose to embed a recently introduced method into the UCR suite procedure, in order to make it even faster. Specifically, we adapt the DTW with Pruned Warping Paths (SILVA; BATISTA, 2016b) to improve the internal efficiency of the DTW calculation. In this way, we can speed up the bottleneck of the similarity search under the warping distance, i.e., the comparison of pairs of time series which the pruning procedure was not capable to discard.

We demonstrate that our method is faster than the proposed by Rakthanmanon *et al.* (2012), considered the fastest tool for the exact similarity search under DTW. The speedup

achieved by our method depends on two factors: (i) the length of the query; and (ii) the total amount of allowed warping. We demonstrate that the runtime of our method is similar to the state-of-the-art for small queries and narrow warping constraints. However, our method performs up to 5 times faster for long queries and large warping windows.

The remainder of this paper is organized as follows. Section 4.2 introduces the notation as well as basic concepts and definitions on time series and the DTW measure. Section 4.3 presents the UCR suite for similarity search. Section 4.4 describes the DTW with Pruned Warping Paths method and how we adapt it to the similarity search procedure. Next, Section 4.5 presents the experimental evaluation to verify the efficiency of our method. Because our method performs better on long queries and a large amount of allowed warping, Section 4.6 discusses the need for both assumptions on several application domains. Section 4.7 introduces how we can adapt the proposed ideas to other distance measures. Finally, Section 4.8 concludes this work.

4.2 Background and Definitions

In this section, we define the basic concepts related to our work and introduce the notation used in the remaining of the paper. We begin by defining a *time series*.

Definition A time series x is a set of N ordered values such that $x = (x_1, x_2, \dots, x_N)$ and $x_i \in \mathbb{R}$. Each value x_i is referred to as an *observation* and N is the *length* of the time series.

Note that, by this definition, a time series does not necessarily need to be defined in time. The only requirement is the logical order of values which needs to be respected. Furthermore, we assume that the interval between two consecutive observations can be disregarded with no loss of generality. This allows the use of the methods described in this section on sequences of real numbers used to describe shapes, spectral data, and other numerical sequences.

Given the definition of time series, we are in the position to define a subsequence.

Definition A subsequence $x_{i,m}$ is a continuous subset of x of length m starting from the observation i , i.e., $x_{i,m} = (x_i, x_{i+1}, \dots, x_{i+m-1})$, such that $i + m < N$.

The focus of this paper is the task of the subsequence similarity search¹, defined as follows.

Definition Subsequence similarity search is the procedure of finding the nearest neighbor – i.e., the most similar subsequence – of a given query time series y with length m in the long (reference) time series x with length N , such that $m \ll N$.

¹ From this point, we use this definition for the terms *subsequence similarity search* and *similarity search* without any distinction between them.

While describing the DTW and pruning techniques we may use the term *time series* for both objects under comparison, even if one of them is a subsequence of a longer time series. Also, we consider that both the query and the subsequence of the reference time series have the same length (m). We notice that the algorithms discussed in this paper may be easily adapted to the nearest neighbor algorithm in batch datasets, i.e., composed of segmented time series which represent specific events.

The most important decision for similarity search is the distance function used to match the subsequences. Despite the existence of several distance functions, there is strong evidence in the literature that Dynamic Time Warping (DTW) is the most suitable distance measure for the task of finding nearest neighbors in time series data for a multitude of application domains (DING *et al.*, 2008; WANG *et al.*, 2013).

The DTW distance achieves an optimal nonlinear alignment of the observations under certain constraints. Specifically, the DTW between two time series of length n and m is the cost of the optimal (n, m) -warping path between them. Such (n, m) -warping is defined as follows.

Definition An (n, m) -warping path is a sequence $p = (p_1, \dots, p_L)$ with $p_l = (i_l, j_l) \in [1 : n] \times [1 : m]$ for $l \in [1 : L]$ satisfying the following three constraints (MÜLLER, 2007):

- Boundary constraint: $p_1 = (1, 1)$ and $p_L = (n, m)$;
- Monotonicity constraint: $i_1 \leq i_2 \leq \dots \leq i_L$ and $j_1 \leq j_2 \leq \dots \leq j_L$;
- Continuity constraint: $p_{l+1} - p_l \in \{(1, 0), (0, 1), (1, 1)\}$ for $l \in [1 : L - 1]$.

Thus, an (n, m) -warping path is a mapping between elements of the time series x and y assigning the observations x_{i_l} of x with y_{j_l} of y . The total cost $c_p(x, y)$ of an (n, m) -warping path p between two time series x and y with respect to a cost measure c is defined by the Equation 4.1.

$$c_p(x, y) = \sum_{l=1}^L c(x_{i_l}, y_{j_l}) \quad (4.1)$$

The cost measure $c(x_{i_l}, y_{j_l})$ is usually defined by the squared Euclidean distance between the pair of observations. Therefore, every (n, m) -warping path is monotonically increasing given that $c(x_{i_l}, y_{j_l}) \geq 0$.

Finally, we define the optimal (n, m) -warping path.

Definition The optimal (n, m) -warping path p^* is the (n, m) -warping path having minimal cost among all possible (n, m) -warping paths, i.e., $c_{p^*}(x, y) = \min\{c_p(x, y) \mid p \text{ is an } (n, m)\text{-warping path}\}$.

A dynamic programming algorithm can calculate the optimal (n, m) -warping path. Equation 4.2 defines the initial condition of the algorithm to estimate the DTW between two time series x and y with lengths n and m respectively.

$$dtw(i, j) = \begin{cases} \infty, & \text{if } i = 0 \text{ or } j = 0 \\ 0, & \text{if } i = j = 0 \end{cases} \quad (4.2)$$

where $i = 1 \dots n$ and $j = 1 \dots m$. From this, Equation 4.3 defines the recurrence relation of DTW algorithm.

$$dtw(i, j) = c(x_i, y_j) + \min \begin{cases} dtw(i-1, j) \\ dtw(i, j-1) \\ dtw(i-1, j-1) \end{cases} \quad (4.3)$$

The DTW distance is given by the value calculated by $dtw(n, m)$. The described algorithm iteratively fills a cost matrix, which we refer to as *cumulative cost matrix* or just *DTW matrix* from now on. Figure 33 shows an example of the DTW between two subsequences, presenting the DTW matrix and the resulting alignment.

Figure 33 – Given two time series under comparison (*left*), the DTW algorithm calculates a cumulative cost matrix (*center*) in order to find the optimal path in this matrix (highlighted in red). With such a path, it is possible to reconstruct the optimal alignment between the series (*right*)

Source: Elaborated by the author.

A space-efficient implementation of the DTW algorithm may use a two-row vector instead of the cumulative cost matrix. Such optimization is possible because the calculation of a given cell only depends on values calculated in the same and previous rows, reducing the DTW algorithm complexity to $\mathcal{O}(n)$ regarding space. However, reducing the $\mathcal{O}(n^2)$ time complexity is a more difficult matter. To the best of our knowledge, the only way to reduce its complexity is by means of approximations – which do not provide any boundaries for the approximation error – or warping windows (SAKOE; CHIBA, 1978; ITAKURA, 1975).

A warping window, or constraint band, defines the maximum allowed time difference between two matched observations. From the algorithm standpoint, this technique restricts the values that need to be computed to a smaller area around the main diagonal of the matrix. In addition to providing a faster calculation, warping windows usually improves the accuracy of the

similarity search and 1-NN classification (WANG *et al.*, 2013). In this work, we consider the warping windows proposed by Sakoe and Chiba (1978).

Because DTW is a costly distance measure, several papers have proposed techniques to improve the runtime of its calculation. More specifically, most of them are indexing methods, focused on the similarity search procedure. In the next section, we discuss these algorithms and the fastest implementation of the similarity search under DTW known so far.

4.3 The UCR Suite

Given the ubiquity of temporal data, there is a plethora of work on speeding up the subsequence similarity search of time series. However, we concentrate our attention on the work by Rakthanmanon *et al.* (2012), which is the first to expedite time series search to an order of trillions of observations. In this work, the authors describe some of the most important speedup methods and present novel techniques to perform the subsequence similarity search in admissible time. Also, they discuss how to use them together to create the fastest tool for exact time series similarity search under DTW available so far, the *UCR Suite*.

In this section, we briefly describe the techniques used to implement the UCR Suite. We refer the reader interested in more details about each method to Rakthanmanon *et al.* (2012). We note that DTW does not obey the triangle inequality. Therefore, indexing algorithms for metric spaces are not applicable to speeding up similarity search under this distance measure.

Given that we are interested in a single nearest neighbor of a given query, we can store the true DTW distance to the nearest subsequence up to a certain moment during the search into a variable *best-so-far* (*bsf*). The main purpose of having a *bsf* is to avoid the expensive DTW algorithm, discarding its calculation for subsequences that are certainly not the best match. In other words, with this value we can restrict the space of nearest neighbor candidates. Despite this approach not explicitly using indexing structures, we refer to the methods that limit the space of candidates as indexing methods or indexing techniques.

The most popular techniques to index time series are lower bound (LB) distance functions. An $LB(x,y)$ function returns a value that is certainly lower or equal to the true $DTW(x,y)$ distance between two time series objects x and y . If such LB is greater than the *bsf*, we know that x is not the nearest neighbor of y . Therefore, the subsequence x can be discarded. Despite the fact that we focus on the nearest neighbor search, this method can be trivially extended to the k -nearest neighbor search by defining the *bsf* as the distance to the k -th nearest subsequence so far.

An LB function needs to fulfill the following requirements to be efficient: (i) its calculation must be fast; and (ii) it needs to be tight, i.e., its value needs to be close to the true DTW. These requirements usually imply in a trade-off between tightness and time efficiency. In general, tight LB functions tend to be more expensive to calculate.

For this reason, [Rakthanmanon et al. \(2012\)](#) proposed to cascade LB functions. The similarity search sorts the LB functions by increasing runtime costs. If the first (and fastest) LB function fails to prune the DTW calculation, then the method tries the next one. If all LB functions fail to prune the DTW calculation, then the method computes the true DTW distance.

There exist several LB functions in the literature. However, the UCR Suite uses only three of them. The authors argue that these three functions subsume all other lower bound measures concerning the tightness-efficiency trade-off. In other words, there is always a faster-to-compute LB with, at least, a similar pruning power.

The first LB calculated in the UCR Suite is the LB_{KimFL} – a simplification of the LB_{Kim} ([KIM; PARK; CHU, 2001](#)) –, which is the sum of the distances between the first and the last pairs of observations of the time series. This measure is guaranteed to be an LB thanks to the boundary constraint of the DTW, specified in Definition 4.2.

The calculation of LB_{KimFL} is extremely fast ($\mathcal{O}(1)$); however, it prunes a small percentage of the DTW comparisons. In the cases LB_{KimFL} fails to prune the distance calculation, the UCR suite uses the LB_{Keogh} ([KEOGH; RATANAMAHATANA, 2005](#)) lower bound function.

Briefly, the LB_{Keogh} constructs an envelope around the query, limited by the minimum and maximum values in the warping window for each observation. The lower bound measure is the squared Euclidean distance between the reference subsequence and the nearest envelope for each observation. LB_{Keogh} is slower (specifically, $\mathcal{O}(n)$) than LB_{KimFL} ; however, LB_{Keogh} can prune a much larger number of objects.

Figure 34 illustrates the above described LB functions.

Finally, if the LB_{Keogh} also fails in pruning the DTW calculation, it is repeated, but inverting the roles of the query and the reference subsequence with respect to the envelope. Such procedure is valid because the LB_{Keogh} is asymmetric. So, its calculation using the reference subsequence to construct the envelope may result in a higher – and consequently tighter – value. In this case, the algorithm changes the value of the LB by the maximum between the two LB_{Keogh} calculated and re-evaluates the pruning of the pair.

In addition to pruning by LB, an important technique to speed up the time series search is early abandoning. In several cases, it is possible to know if the distance between a pair of time series will be greater than the bsf during the computation of necessary values for the search procedure. One example is the early abandoning during the LB_{Keogh} calculation. While calculating the LB, we incrementally increase its value. If at any step we find that the value of the partial LB is greater than the bsf , we can stop its calculation and skip to the next subsequence in the similarity search.

Two techniques may be used in this step to further improve the runtime of the search in addition to the LB calculation. The first one is z-normalization. The normalization procedure is necessary to improve the matching of subsequences in the presence of offset and amplitude

Figure 34 – The LB_{KimFL} (left) estimates a lower bound as the sum of the Euclidean distances of the first and last pairs of observations. The LB_{Keogh} (right) constructs an envelope around one of the time series and estimates the lower bound as the Euclidean distance from the other sequence to the nearest (lower or upper) envelope for each point outside the region encapsulated by the envelope

Source: Elaborated by the author.

variation [Keogh et al. \(2009\)](#). This procedure transforms the time series such that the mean of its observations is $\mu = 0$ and the standard deviation is $\sigma = 1$.

A straightforward batch algorithm to calculate the z-normalization can harm to the search runtime. Instead, the UCR suite implements it incrementally. This approach allows interspersing the z-normalization with the LB calculation. In this way, if we can early abandon the LB function, we are also abandoning the z-normalization calculation.

For this purpose, we need to look at the definitions of mean and squared standard deviation (variance). Equation 4.4 defines such statistics for a subsequence of the time series x starting at the p -th observation.

$$\mu = \frac{1}{m} \left(\sum_{i=p}^{m-1} x_i \right) \quad \sigma^2 = \frac{1}{m} \left(\sum_{i=p}^{m-1} x_i^2 \right) - \mu^2 \quad (4.4)$$

Given the mean and the variance of $x_{p,m}$, these values may be reused in the calculation of the statistics referring to the subsequence $x_{p+1,m}$. This is done by subtracting the observation x_p and adding x_m to the summations. In the case of the standard deviation, this procedure uses the squared value from the observations. Once we keep the sum and the squared sum of the observation from a subsequence, we can update the mean and standard deviation to the next subsequence in a constant time.

The second additional method is the reordering of observations to calculate the LB. Instead of calculating the LB in the *natural* order (from the first to the last observation), we may

sort the calculation by the absolute value of the query. This simple modification is likely to lead to the early-abandon of the LB calculation in fewer steps.

Finally, if all the previously described methods were not enough to avoid the DTW calculation, it is still possible to not calculate the whole cumulative cost matrix. Specifically, we can early abandon the DTW calculation when the minimum value obtained in a row (or column) of its cost matrix is greater than the *bsf*. In this case, the monotonicity property of DTW (c.f. Equation 4.1) guarantees that the final value is also greater than the *bsf*. We can use the information of the partials calculated by the lower bound function to improve the distance early abandoning. Consider we are storing the cumulative lower bound from each point to the end of the time series. After the calculation of each row i of the DTW matrix, we can estimate a new LB of the final distance between the time series x and y given by $DTW(x_{1,i}, y_{1,i}) + LB(x_{i+1,m-i}, y_{i+1,m-i})$. So, if such a value is greater than the *bsf*, the distance computation can be abandoned. Figure 35 summarizes the similarity search techniques implemented in the UCR Suite.

Figure 35 – The UCR Suite sequentially applies different methods to avoid the costly DTW computation. The distance calculation to the current subsequence may be abandoned at any step (dashed lines). The z-normalization and the LB_{Keogh} are calculated at the same time (third box), so both can be early abandoned together. DTW is calculated only if the LB and early abandon methods were not successful in pruning it. The value of the *bsf* is updated accordingly before the search continues to next subsequence. This process continues until the last subsequence is assessed

Source: Elaborated by the author.

We performed an experiment to measure the runtime of the indexing techniques used by the UCR Suite and the time taken by the DTW calculations. For this purpose, we measured the time to search a query of length 256 in an electrocardiography time series (c.f. Section 4.5.1.3). The query was randomly selected from the data.

Fixing a relative warping window of 10% of the query length, the percentage of DTW calculations was 1.42% of the total number of assessed subsequences. Despite the incredibly small amount of DTW calculations, the relative time to calculate them (even with the early abandoning) corresponded to approximately 25% of the runtime.

This is even more evident for longer queries or larger warping windows. For instance, using the same data with a relative warping window of 20% of the query length, the number

of DTW calculation was 4.06% of the total number of subsequences. The time to calculate the distances took approximately 60% of the whole search procedure runtime.

In this paper, we improve the UCR Suite by adapting a recently introduced algorithm, proposed to speed up DTW calculations independently of the *bsf*, named DTW with Pruned Warping Paths (SILVA; BATISTA, 2016b). We extended our previous work incorporating the *bsf* to improve its performance in the similarity search. In the next section, we describe this method, as well as its adaptation to the similarity search task.

4.4 DTW with Pruned Warping Paths

In this paper, we improve the UCR suite performance by augmenting it with a recently proposed method called DTW with Pruned Warping Paths – PrunedDTW – (SILVA; BATISTA, 2016b). PrunedDTW was introduced as an alternative to speed up DTW calculations when the use of indexing methods is not applicable, such as applications that require the all-pairwise distance matrix within a set of time series. One example is the widely known family of hierarchical clustering (XU; WUNSCH, 2008), given that most of these algorithms require the relation among all the objects in the data set.

Although we have proposed PrunedDTW for a scenario in which current indexing techniques are not applicable, we can adapt it to similarity search. PrunedDTW is orthogonal to all lower bound functions and other indexing-based algorithms in the time series literature. In other words, the application of PrunedDTW on the similarity search is complementary to any adopted indexing techniques. While most methods to speed up the time series similarity search “compete” between them, PrunedDTW explores an entirely different strategy.

In this scenario, we propose the use of PrunedDTW when all the evidence obtained by the indexing methods was not sufficient to reject the costly dynamic programming-based distance calculation. Also, we introduce a subtle modification of PrunedDTW to use the *best-so-far* distance to improve its performance. Before we explain this change, we introduce the PrunedDTW algorithm.

4.4.1 The Intuition Behind PrunedDTW and its Pruning Strategies

Figure 36 provides a heat map of the DTW matrix for two electrocardiography time series. Note that the values around the main diagonal are relatively close to the actual DTW distance, which is approximately 2.77 in this case. In contrast, most of the values in the matrix are much higher than the real distance.

The value in each cell (i, j) of this matrix represents the cost of the best alignment between the subsequences $x_{1,i}$ and $y_{1,j}$. Any alignment between x and y which contains such partial alignment has a total cost that is greater or equal to the value stored in the cell (i, j) . In

Figure 36 – DTW matrix between two electrocardiogram subsequences. The colors indicate the value obtained in each cell of the matrix

Source: Elaborated by the author.

the case in which such cell contains a value that is greater or equal than the actual distance, the optimal partial alignment ending by matching the pair (i, j) is guaranteed to not belong to the optimal path between the whole time series under comparison. Therefore, the DTW algorithm can skip the calculations of all the alignments that contain such partial.

To use the observed fact to speed up DTW calculations, PrunedDTW works with a distance threshold to determine if a cell is amenable to pruning. For this, the original proposal uses the squared Euclidean distance (ED) as an upper bound (UB) to DTW, i.e., a value that is guaranteed to be greater or equal the actual distance. So, any cell that has a value greater than the UB can be pruned, because it is guaranteed to not belong to the optimal warping path.

Also, the algorithm uses such threshold to establish pruning strategies to decide when to start and finish the computations in each row² of the DTW matrix. Figure 37 exemplifies the pruning approach, that relies on monitoring two variables: the starting column sc and the ending column ec .

In the row $i + 1$, the first two columns have a value greater than UB. Therefore, the variable sc is set to column 2 and the processing can safely start at column 2 in the next row. We can prune the computation of the cells containing A and B thanks to the values used by the DTW recurrence relation represented by the three arrows. Given the initialization with infinity and the large value already calculated in the cell $(i + 1, 0)$, the variable A will obligatorily have a value greater than UB. The same occurs to B which depends on $A > UB$ and other two cells in the previous row also greater than UB. In contrast, the cell with the value C may have a value smaller than UB since it depends on the cell $(i + 1, 2) \leq UB$.

² Our implementation traverses the matrix in row-major order. However, the algorithm can also be implemented by traversing the matrix in column-major order.

Figure 37 – Strategies adopted by PrunedDTW to prune the beginning and the end of each row of the DTW cumulative cost matrix. The variable sc (left) denotes the position of the first value lower or equal than the UB in the previous row and is used as the index to start the current iteration. On the opposite, the variable ec (right) points to the first value in the previous row that is greater than the UB. From this point, the algorithm may stop the calculation as soon as it finds a value that is greater than the UB

Source: Elaborated by the author.

The initial value of the variable sc is 0, i.e., while no values greater than UB are found, the calculation in each row will start at the first column. In the case that a warping window is used, the rows will be initiated in the column with the highest index between the column established by the pruning criteria and the one related to the warping window.

The second pruning strategy is responsible for pruning the last columns of the current row. The variable ec points to the first of a continuous sequence of values greater than UB that finishes at the end of the row. This value defines the column where we can stop the calculation of the next row.

In the presented example, row $i+1$ is processed and ec is set to $j+2$. We can stop the row $i+2$ as soon as two criteria are met: (i) the last calculated value is greater than UB and (ii) the current column index is greater or equal to ec . Suppose that A is greater than UB. In this case, criterion (ii) is not met. We can see that B may be smaller than UB since it can use $(i+1, j+1)$ which is lower or equal to UB. However, if B is greater than UB then both criteria are met, and we can stop processing the current row. This occurs because C and D can only inherit values from the matrix that are greater than UB.

For further details about the original PrunedDTW method including algorithm and detailed performance results for the problem of computing the all-pairwise distance matrix, we refer the reader to (SILVA; BATISTA, 2016b). In this paper, we propose a subtle variation of PrunedDTW which can significantly improve the performance of existing similarity search indexing methods. For clarity, we refer to our proposal as SS-PrunedDTW (for Similarity Search PrunedDTW).

4.4.2 Embedding PrunedDTW into the Similarity Search Procedure

The results presented by Silva and Batista (2016b) demonstrate that PrunedDTW can speed up the traditional all-pairwise DTW distance calculation from two to ten times. Such a variance is the result of the tightness of the adopted UB function – Euclidean distance. The

tightness of an upper bound is related to how close its values are from the actual DTW distance, which may fluctuate between different datasets. In general, a tight UB allows PrunedDTW to prune a large number of calculations, achieving a higher speedup.

Euclidean distance is a natural and efficient UB for DTW. ED is the DTW distance obtained by the warping path defined by the main diagonal of the DTW matrix. As DTW returns an optimal path, i.e., the path that leads to the smallest distance, the DTW between two time series is always lower or equal than ED. Although ED is a reasonably tight UB for DTW, in similarity search we have access to a potentially much tighter UB, the *best-so-far* distance.

Note that the *bsf* is not an upper bound for the DTW distance between the query and the subsequence from the long time series. Instead, it is an upper limit to consider such subsequence as the (k -th) nearest neighbor of the query. Following the same principle of the distance early abandoning strategy, any partial alignment with a value greater than the *bsf* leads to a DTW distance greater than the distance to the current (k -th) nearest neighbor. So, the *bsf* is an admissible threshold for pruning in the similarity search scenario. Also, the *bsf* has the advantage that its value is monotonically decreasing during the search.

Regarding pruning power, *bsf* is usually much smaller than the ED between two time series under comparison. Then, using the *bsf* must imply in a higher number of skipped calculations. The *bsf* is usually smaller than the DTW distance between the two subsequences, and a simple observation can help us to understand why. While the DTW between two arbitrary subsequences can vary widely, the *bsf* is the DTW distance between the two closest time series compared until a certain point of the search process, which is independent of the current pair of subsequences.

Figure 38 visualizes the behavior of the DTW, ED, and *bsf* in a similarity search on a dataset of electrocardiography (c.f. Section 4.5.1.3). In this case, we stored the ED and *bsf* every time that we needed to calculate the DTW, i.e. when the lower bound functions were not able to prune the candidate for nearest neighbor. As we can see, the value of the *bsf* is usually much lower than the ED, except in the first distance calculation – when the *bsf* is still undefined and, then, initialized as infinite. In contrast, the ED depends only on the two time series under comparison, so its value has a significant fluctuation during the search process.

The experiment presented in Figure 38 indicates that the *bsf* is a better threshold for pruning decisions in PrunedDTW than the ED. To verify this statement, we experimented with all the datasets used in our experimental evaluation (c.f. Section 6.5). For each dataset, we established ten distinct search scenarios, varying the query and warping window lengths. One characteristic is common for every experimented scenario: after the first DTW calculation, when *bsf* is infinite, in 100% of the subsequences not discarded by the pruning techniques, the *bsf* is lower than the ED between these subsequences. Specifically, the *bsf* is approximately 7.77 times lower than the ED in average. Besides, the *bsf* stores a value that is lower than the actual DTW in 94.14% of these cases.

Figure 38 – Comparison of the Euclidean distance, Dynamic Time Warping, and best-so-far values during the similarity search. Specifically, this figure visualizes the distances obtained each time that the DTW was calculated in a search for a 128 observations long query and a relative warping window size of 10% of the query length in an electrocardiography dataset. We highlight a moment of the search in which the current DTW is lower than the *bsf*, updated for the next iteration

Source: Elaborated by the author.

As an additional feature, just like in the distance early abandoning, we may improve PrunedDTW using partial lower bound calculations. When evaluating if a cell is liable for pruning, PrunedDTW originally considers only the value obtained by the recurrence relation of the DTW algorithm for that cell.

Alternatively, we can sum such value to the cumulative lower bound from the first observation ahead of the values comprised by the warping window to the end of the subsequences. In other words, we can use the total cost of the partial alignment summed to such lower bound partials and compare to the *bsf* in order to decide the pruning. For clarity, consider that $cumLB[i]$ stores the summed contributions of LB_{Keogh} from the i -th position to the end of the envelopes (c.f. Figure 34-right). For a value stored in $D[i, j]$, i.e., from the partial alignment ending in the i -th and j -th observations of the subsequences x and y , we guarantee that $DTW(x, y) \geq D[i, j] + cumLB[i + i + ws] \geq D[i, j]$, where ws is the absolute warping window length. Once we only have interest in a pair of subsequences x and y if $DTW(x, y) < bsf$, we need to have as result that $bsf > D[i, j] + cumLB[i + i + ws]$. For this reason, we can use the *bsf* and the cumulative LB as an upper bound. Specifically, if any partial alignment is such that $bsf - cumLB[i + i + ws] < D[i, j]$, we guarantee that this alignment will not lead to a distance value lower than the *bsf*.

We are now in a position to describe the algorithm in details. Algorithm 6 implements the SS-PrunedDTW using $\mathcal{O}(n)$ space. Note that, for simplicity, we omitted the early abandoning of the DTW distance calculation.

The algorithm starts by defining auxiliary variables to the pruning strategy (lines 4 to 6) and by setting the initial values of the cumulative cost matrix (lines 7 to 9).

Algorithm 6 – SS-PrunedDTW algorithm**Requer:** Time series x and y , with length N 1: Warping window size ws 2: Best-so-far distance bsf 3: Cumulative LB values $cumulativeLB$ with the LB for each subsequence**Assesure:** The distance between x and y according DTW

▷ Auxiliary variables to prune decisions

4: $sc \leftarrow 0$ 5: $ec \leftarrow 0$ 6: $lp \leftarrow 0$

▷ last pruning control

▷ Initialize the vector of DTW calculations of the previous row

7: **para** $i \leftarrow 1$ **até** N **faça**8: $D_{prev}[i] \leftarrow \infty$ 9: **fim para**10: **para** $i \leftarrow 0$ **até** $N - 1$ **faça**11: $smaller_found \leftarrow FALSE$ 12: $pruned_ec \leftarrow FALSE$ 13: $ec_next \leftarrow i$ 14: $ub \leftarrow bsf - cumulativeLB[i + ws + 1]$ 15: **para** $j \leftarrow \max(0, sc, i - ws)$ **até** $\min(i + ws, N - 1)$ **faça**16: **se** $(j = 0 \text{ and } i = 0)$ **então**

▷ first cell in the cumulative matrix

17: $D[0] \leftarrow sqED(x_0, y_0)$ 18: $min_cost \leftarrow D[0]$ 19: **se** $D[0] \leq ub$ **então**20: $smaller_found \leftarrow TRUE$ 21: **fim se**22: **continue**

▷ skip the remaining of this loop

23: **fim se**24: **se** $j \geq lp$ **então**

▷ avoid garbage at the end of the row

25: $D_{prev}[j] = \infty$ 26: **se** $j > lp$ **então**27: $D_{prev}[j - 1] = \infty$ 28: **fim se**29: **fim se**30: $D[j] = sqED(x_i, y_j) + \min(D[j - 1], D_{prev}[j], D_{prev}[j - 1])$ 31: **se** $D[j] > ub$ **então**

▷ pruning strategy

32: **se** $j \geq ec$ **então**33: $lp \leftarrow j$ 34: $pruned_ec \leftarrow TRUE$ 35: **break**

▷ break the for loop / jump to the next row

36: **fim se**37: **senão**38: **se** $smaller_found = FALSE$ **então**39: $sc \leftarrow j$ 40: $smaller_found \leftarrow TRUE$ 41: **fim se**42: $ec_next \leftarrow j + 1$ 43: **fim se**44: **fim para**45: $D_{prev} \leftarrow D$

▷ Pruning information updates

46: **se** $pruned_ec = FALSE$ **então**47: $lp \leftarrow i + 1 + ws$ 48: **fim se**49: $ec \leftarrow ec_next$ 50: **fim para**51: **se** [**então** last row was pruned] $pruned_ec = TRUE$ 52: $D[N] \leftarrow \infty$ 53: **fim se****retorna** $D[N]$

The *for* loop from line 7 to 47 traverses the observations of the series x . It starts by defining the initial values of the pruning-related variables for the current iteration (lines 11 to 14).

The next *for* loop traverses the observations of the time series y constrained by the warping window, which length is defined by ws . In the first time that the algorithm achieves this point, it is necessary to set the first value in the cumulative cost matrix. It is done by the settings inside the condition starting in line 16 (which finishes at line 23). This condition is necessary to perform a correct initialization of such a structure.

When implemented to use linear space, the pruning of the last values in a row may cause problems with non-computed cells in the cumulative cost matrix. It occurs when the last row is pruned, and the cell which should contain the distance is currently storing a value from a previous iteration. In this case, we only need to check in which column the last row was pruned and avoid the values stored in any column ahead of it. This is done by the condition between lines 24 and 29.

Next, we calculate the value of the current cell of the DTW matrix, in line 30. Notice that we simplified this line for the sake of presentation. In an actual implementation of the algorithm, it is necessary to check if $j - 1$ corresponds to a valid index. For clarity, $j - 1$ may never be lower than its initial value defined in the heading of the internal *for* loop, i.e., $\max(0, sc, i - ws)$. In this case, we use infinity instead of the partials $D[j - 1]$ and $D_{prev}[j - 1]$.

This step of the algorithm finishes by checking whether the current row shall be pruned and if any information related to the pruning mechanism needs to be updated (lines 31 to 43). It first checks if the current value is greater or equal the bsf . In this case, it is possible to prune the end of the row, regarding only one more condition. Specifically, if the index of the current column is greater or equal the ec (line 32), then we store this index in the variable lp (line 33). Afterward, we mark the row as pruned (line 34) and prune the row calculation by skipping the next iterations on this row (line 35).

In the case that the current value is lower than the bsf , we need to update the values sc (line 39, case it was not set in this row yet) and ec_{next} (line 42), which is an auxiliary variable to set the ec for the next row.

After finishing the internal *for* loop, we first set the vector used as the previous row in the next iteration with the values of the currently calculated row (line 45). Then, we update the variables related to the pruning for the next row (lines 46 to 49). In the case that no pruning occurred at the end of the row, we set the variable lp to the index related to the last column of the warping window in the next row (line 47). In addition, we set the variable ec as ec_{next} (line 49).

Finally, we return the final distance value. However, we first check if the end of the last row was pruned (lines 51 to 53). In this case, we force the algorithm to return infinity (or any value indicating early abandoning of the distance calculation)

4.4.3 On the Correctness of the SS-PrunedDTW

Consider $P = \{p_1, p_2, \dots, p_k\}$ the set of all k possible (n, m) -warping paths between two temporal data in which p_o denotes the optimal (n, m) -warping path. By definition, any $p_i | i \neq o$ has a cost greater or equal to the cost of p_o . Although unlike, there may be other (n, m) -warping paths with costs equal to the optimal alignment. However, most of the paths p_i do not fit this circumstance and may be disregarded.

The cell (i, j) in the cumulative cost matrix stores the cost of the optimal (i, j) -warping path, i.e., the optimal alignment between the subsequences $x_{1,i}$ and $y_{1,j}$. Recall that the cost of matching two observations is nonnegative. Thus, any (n, m) -warping path containing the (i, j) -warping path has a cost that is at least the value stored in the cell (i, j) . If this value is greater than the cost associated to p_o , so the (i, j) -warping path is not part of the optimal alignment between the time series under comparison.

When such observations are embedded in a similarity search scenario, the notion of which warping paths are relevant is now associated with the threshold defined by the *bsf* and the cumulative lower bound. Specifically, any (i, j) -warping path with a cost greater than the specified UB may be ignored. For this reason, we may say, without loss of generality, that SS-PrunedDTW is based on the same premise as the early abandoning of the distance calculation.

The correctness of the pruning strategies can be obtained by the ordering of the fulfillment of the DTW matrix. According to Equation 4.3, the value of each cell is influenced by three other values:

- (I) same row, preceding column: $(i, j - 1)$;
- (II) preceding row, same column: $(i - 1, j)$;
- (III) preceding row and column: $(i - 1, j - 1)$.

For the pruning strategy that determines the value of the starting column variable (*sc*), our proposal can be better understood if we observe how the cells are calculated since the column zero of the DTW matrix. Because such column is initialized with infinity, the value inherited from (I) never leads to the minimal value for the first column in the DTW matrix. This fact is likewise true for any cell that starts the calculation in a row – determined by pruning or warping constraints. Similarly, while the currently calculated value exceeds the UB, any value obtained by (I) is also greater than the distance to the $(k - th)$ nearest neighbor – what only occurs due to (II) or (III). In any case, such a value could be admissibly pruned.

The analysis of the values in (II) and (III) becomes necessary from the column in which there is a value lower than the UB in the preceding row. Therefore, while the values of a row are calculated, our algorithm stores the position where this value occurs for the first time and uses this information to start the next row. In other words, our method does not prune the calculation by determining the beginning of a row in a column c if there is at least one promising value in

the preceding row in any column $c' < c$. Thus, it is guaranteed that our method does not miss any promising value in (II) or (III).

The two restrictions used to define the pruning strategy of the end column in each row of the matrix guarantee the correctness of our method by the following facts. The calculation of the values in a row will never be pruned while the current value is lower than the UB, ensuring that there will be no missed promising values at the positions defined by (I). Also, we ensure that there is no loss of promising values in (II) and (III) by the fact that the algorithm monitors, with the variable ec , from which point there are no more promising values in the preceding row. A row can only be pruned if its current column is greater than ec . In other words, when calculating a value for a column c , this criterion requires that $ec \leq c - 1$.

4.5 Experimental Evaluation

Given that we have presented our algorithm and proved its correctness, we evaluate the effect of SS-PrunedDTW in improving the runtime of the similarity search. For this purpose, we have modified the UCR suite and executed the same experiments using both implementations. For clarity, we refer to the proposed implementation as UCR-USP suite.

We note that we have made available the source code, as well as detailed results, in a supplementary website ([SILVA et al., 2016](#)).

In our experimental evaluation, we used 6 datasets from different application domains. For each of them, we defined a long reference time series and five different queries. We do not use any knowledge from experts to define the queries. To create the queries, we randomly picked subsequences from a time series that was not used to compose the reference data. For each dataset, we selected five different queries. Also, we vary the length of the query. For this, we cropped each selected query in lengths of 128, 256, 512 and 1024. Finally, we used five different widths of the warping window: 10%, 20%, 30%, 40%, and 50% of the length of the query. So, we have experimented with 100 different search scenarios for each dataset.

For simplicity, we only evaluated the methods in single-dimensional time series. However, we note that the generalization of the applied algorithms to the multidimensional is simple. For a detailed discussion on the generalization of DTW to deal with multi-dimensional data, we refer the reader to [Shokoohi-Yekta et al. \(2017\)](#) and [Górecki and Łuczak \(2015\)](#).

Finally, we ran all the experiments on the same computer³. To avoid spurious time fluctuations, we guaranteed that, at any time, there was only one process – except OS processes – running on the computer.

³ The experiments were carried out in a desktop computer with 12 Intel(R) Core(TM) i7 – 3930K CPU @ 3.20GHz and 64Gb of memory running Debian GNU/Linux 7.3.

4.5.1 Data

We start the description of our evaluation by briefly presenting the datasets, regarding the application domain and the length of the reference time series.

4.5.1.1 Physical Activity Monitoring

Due to the increasing availability of sensors, such as the accelerometers present in the majority of smartphones, the human activity monitoring is an application with a growing attention. In this work, we use the dataset PAMAP2 (REISS; STRICKER, 2012), which contains recordings of 18 different activities performed by 9 subjects. The data have a sampling rate of 100Hz . In our experiments, we used the time series obtained by z -axis measurements from the accelerometer in the arm position. The reference time series has 3,657,033 observations.

Figure 39 shows how the accelerometers are arranged and an example of the collected data.

Figure 39 – The PAMAP data are collected from three different accelerometers (*left*), positioned according the red circles. We used the z -axis of the accelerometer in the hand position. The presented time series (*right*) was obtained by two seconds of walking

Source: Elaborated by the author.

4.5.1.2 Athletic Performance Monitoring

Monitoring activity may also be used for professional ends. One such example is the tracking of athletes' performance. For this, the athletes may wear sensors for recording speed, trajectory, energy and other features. In this work, we used the position in the axis attack/defense recorded by ZXY Wearable Tracking sensors⁴ in several soccer players during three matches (PETTERSEN *et al.*, 2014). The data has a sampling rate of 20Hz . The reference time series has 1,998,606 data points obtained by concatenating data from the first two matches from a single player. We randomly chose the queries from the third game.

4.5.1.3 Electrocardiography

Time series are a common category of data on health applications for a long time. One of such applications is the monitoring of heart activity by electrodes placed in the skin, a procedure known as electrocardiography (ECG). To evaluate our method in this type of signal, we used

⁴ <http://chyronhego.com/sports-data/zxy>

the MIT-BIH Arrhythmia Database (MOODY; MARK, 2001; GOLDBERGER *et al.*, 2000), a collection of 48 ECG recordings digitized at 360 samples per second. For clarity, two seconds of data (720 data points) may contain information of approximately three beats. Our reference time series for this dataset is composed of 27,950,000 values.

4.5.1.4 Photoplethysmography

Another application related to health care evaluated in this work is the photoplethysmography (PPG). This technique is a non-invasive alternative for monitoring the heart rate and cardiac cycle. An optical sensor placed in a peripheral portion of the patient's body generates the data. In this work, we use PPG data collected from the fingertip (KACHUEE *et al.*, 2015). The number of observations in the reference time series is 333,570,000, with a sampling rate of 125 samples per second.

Figure 40 shows an example of the optical sensor used for the monitoring and five seconds of a PPG data.

Figure 40 – A fingertip oximeter (*left*) and five seconds of PPG data obtained by its use (*right*)

Source: Elaborated by the author.

4.5.1.5 Freezing of Gait

The last medical application employed in this work is the detection of freezing of gait (FoG), a symptom related to the Parkinson's disease. In this work, we performed the similarity search in the Daphnet FoG data (BACHLIN *et al.*, 2010). Specifically, we used recordings of the horizontal forward acceleration of the subject's thigh as data. This is the smallest experimented dataset, being that the reference time series contains 1,724,584 observations.

Figure 41 shows an example of the time series in the transition from a normal state to a FoG episode.

4.5.1.6 Electrical Load Measurements

Time series data from measuring the electrical consumption has attracted the attention of researchers because of its wide range of applications. Some examples are the smart energy services (such as automatic demand response) and smart home and smart city solutions. The REFIT dataset (MURRAY *et al.*, 2015) is composed of the electrical consumption monitoring (in Watts) of distinct appliances from 20 different households. The data are sampled such that

Figure 41 – This subsequence of the acceleration of a subject’s thigh shows the transition from a normal state (with high amplitude) to a freezing of gate episode (where the amplitude is clearly lower)

Source: Elaborated by the author.

the interval between observations is 8 seconds. The version of this dataset used in this work was cleaned to avoid missing data, which was substituted by the value zero. To prevent division-by-zero during the z-normalization, we have added a small amplitude noise to the original signals. In our experiments, we used the monitoring of dishwashers, composing a reference time series of 78,596,631 observations.

4.5.2 Results and Discussion

SS-PrunedDTW is an exact algorithm that does not lead to false dismissals. Therefore, the UCR-USP suite provides the same answers as the UCR-suite or any other exact approach based on DTW. Therefore, we perform our evaluation comparing only the runtime of UCR-USP and the UCR suites. We compare the runtime of these two methods varying two parameters: query length and warping window length. We measure the warping window length by the number of observations in the warping window.

Figure 42 shows the runtime of the UCR and the UCR-USP suites for each dataset according to the query length.

The other parameter which directly affects the results is the warping window length. Figure 43 shows the difference in runtime between the experimented suites when this parameter is varied.

The UCR-USP suite outperformed the UCR suite for most of the settings in our experiments. In the few cases in which it did not happen, the difference between the two methods is considerably small. Even more important, the UCR-USP suite only achieved similar or slightly worse performance in the cases with the smallest runtime among our experiments. Specifically, in the worst case – concerning search runtime – in which our method performed slightly slower in average occurred for the dataset of PPG and with a query containing 256 observations and the warping window size of 10%, i.e., with an absolute warping window comprising 25 values. In this case, the search lasted 86.4 seconds, while the UCR suite took 86.1 seconds. On the other hand, with the same dataset, but applying an absolute window size of 512, our method

Figure 42 – Runtime of both UCR and UCR-USP suites on the experimented datasets by varying the query length. The datasets are presented sorted by the length of the reference time series. The plotted values represent the average runtime over 5 different queries per query length

reduced the average runtime from approximately 116,100 to 22,250 seconds, i.e., the UCR-USP suite performed 5 times faster. In general, the results show that the slower the similarity search procedure, the higher the improvement provided by the UCR-USP suite.

To illustrate this fact, Figure 44 shows the speedup ratio ordered by total time taken by the UCR suite in the ECG dataset. The speedup ratio is the runtime of the UCR suite divided by the runtime of the UCR-USP suite. In other words, a speedup ratio of 2.0 means that our method is two times faster in that experiment.

Given the presented results, we recommend the use of UCR-USP suite for all settings. However, we strengthen that its use is remarkably recommended in cases where long queries or large warping windows are required. In fact, the largest absolute warping window in which

Figure 43 – Runtime of both UCR and UCR-USP suites on the experimented datasets by varying the (relative) warping window length. The datasets are presented sorted by the length of the reference time series. The plotted values represent the average runtime over 5 different queries per warping window length

the UCR suite performed better than our proposal – in the average of different queries –, was composed of 25 observations (10% of a query with 256 data points). For all larger absolute warping windows, our method outperformed the UCR suite.

Our results on larger queries and window sizes are sound. The larger the DTW matrix is, the longer it will take calculate it. Therefore, large DTW matrices allow more opportunities for pruning and speedup for SS-PrunedDTW. We notice that several applications may require long queries and large warping windows. For this reason, we dedicate the next section to discuss such subject.

Figure 44 – Speedup ratio in the ECG dataset. There is an increasing trend according to the total time taken by the UCR suite, i.e., the speedup is inversely proportional to the runtime of the original suite. This trend is similar in all evaluated datasets. The markers represent real runtime measurements, and the dashed line presents an exponential trend line for such values

Source: Elaborated by the author.

4.6 On the Need of Long Queries and Large Warping Windows

The results presented in the previous section show that the improvements in performance provided by the UCR-USP suite are more significant for long queries and large warping windows. In this section, we show that these characteristics are not only likely to appear, but *required* in some cases.

This scenario is particularly interesting because it defines the worst case of the similarity search under DTW. In this case, the LB-based pruning strategies tend to be less effective, i.e., the search procedure will require a higher number of DTW calculations. Also, the cost to calculate each DTW distance is also higher.

Long queries are becoming more often with newer technologies. For instance, the recent sensor technologies are more accurate and able to acquire data in higher sampling ratios. In other words, these technologies can obtain time series with a larger number of observations per second. For this reason, short subsequences can only represent a short period. As a consequence, the assessed subsequences need to be even longer than the ones used in our experimental evaluation. This topic is deeper discussed in Section 4.6.1.

At the same time, there is a multitude of applications in which a little warping window may be inadequate. For example, time series from human activities, motion, and locomotion represent people performing activities possibly in completely different paces, resulting in time series with notable differences regarding local and global scaling (SHEN *et al.*, 2017). Another example is the similarity in music applications, such as the query-by-humming (PARK, 2015) and music information retrieval by emotion (DENG; LEUNG, 2015) or rhythm (REN; FAN; MING, 2016). Several other applications may require warping windows larger than the usually adopted 10%. Section 4.6.2 better explores this topic and presents a practical example.

4.6.1 Query Length

We begin by discussing the queries length. The longest query used in our experiments has 1,024 observations. However, we notice that this length is usually too small for several applications.

As an example, consider the dataset REFIT (c.f. Section 4.5.1.6), which stores the measurement of the power consumption every eight seconds. So, a query of 1,024 points corresponds to exact 8,192 seconds, i.e., approximately 2 hours and 15 minutes. If one is interested in analyzing daily consumption patterns, the adequate query should be composed of 10,800 values. Ten thousand data points are one order of magnitude longer than our longest query in the previous experiments. Moreover, some applications may use even longer data. In the example of the power consumption, one may be interested in finding weekly or even monthly patterns, for instance.

While we used the dishwasher consumption in our experiments, Figure 45 shows two 24-hour examples of electric heater consumption monitoring, which brings a more intuitive example. In this example, we used the power monitoring during the Saint Patrick's (March 17th – early Autumn) and the Christmas (December 25th – Winter) days of the year 2014.

Figure 45 – Heater monitoring during 24 hours in different seasons. Despite the similar pattern in the first 8 hours of the day, the daily pattern is clearly different

Source: Elaborated by the author.

Notice that both time series present a similar square shape during dawn. However, this pattern lasts for a little less than 8 hours and a query of 1,024 points (8 hours correspond to 3,600 observations) will not properly represent it.

We performed an experiment using one day long queries in this dataset, i.e., 10,800 observations. Specifically, the reference time series used in this experiment is the power consumption of the heater from only one house. All the queries come from the same kind of device but from another house. The length of the reference time series, in this case, is 6,960,008 values.

Using this dataset, the speedup achieved by the UCR-USP suite varies between 1.12 (when using the relative warping window $w = 0.1$) to a slightly better rate of 1.33 (with $w = 0.5$). Applying our modified suite on the dishwasher dataset with queries composed of the same number of observations, we can notice that our method may achieve better results depending on the data. For the dishwasher dataset, the speedup ratio varied from 1.42 to 3.5 using $w = 0.1$ and $w = 0.5$, respectively. It is important to notice that even if the speedup of 1.42 seems “humble”,

it represents a decay of runtime from 97,763 to 68,679 seconds. In other words, in that case, the UCR-USP suite is saving 29,084 seconds, which represents approximately 8 hours, per searched query.

Another example is the athletic performance monitoring. The sensor generates data with a sampling rate of 20Hz . It means that a query with 1,024 observations corresponds to just 51.2 seconds. Clearly, it is not enough data to analyze the moving pattern of any player. Figure 46 shows an example of the trajectory of a player by time series with lengths 1,024, 6,000 and 12,000 observations, i.e., 51.2 seconds, 5 minutes, and 10 minutes, respectively. For the sake of exemplification, we used the bi-dimensional trajectory time series.

Figure 46 – Examples of trajectories of an attack player monitored during 51.2 seconds (*left*), 5 minutes (*center*), and 10 minutes (*right*). Notice that in the short subsequence it is not possible to observe any playing pattern. In the subsequence regarding 5 minutes, it is possible to note that this player’s positioning is more concentrated in the head of the penalty area. This pattern is more clear in the last case, where it is possible to observe a wide moving pattern close to the middle-field and more concentrated in the attack

Source: Elaborated by the author.

In the case of the soccer data, we analyzed the query lengths: 6,000 and 12,000. For the former case, the speedup ratios varied from 1.22 (using $w = 0.1$) to 3.22 (when $w = 0.5$). In the case of 12,000 observations in the query, – as expected – the speedup was a bit higher, varying between 1.72 and 3.7 (using $w = 0.1$ and $w = 0.5$, respectively).

In both dishwasher and soccer players monitoring datasets, these experiments contribute to clarify the fact that the longer the query, the higher the speedup achieved by our method. For instance, the speedup ratio on the soccer dataset varied between 1.1 and 2.6 when we searched for 1,024 observations long queries.

4.6.2 Warping Window

The warping window is an important parameter of DTW. In some tasks, such as classification, it may cause a significant impact on the results. Empirical evidence has shown that for the UCR Time Series repository datasets (CHEN *et al.*,), small window sizes are more likely to provide superior 1-NN classification accuracy (RATANAMAHATANA; KEOGH, 2005). However, the best value for this parameter is strictly data dependent. There are several datasets in which a large warping window is necessary to achieve the best accuracy results.

One example is the PAMAP data (c.f. Section 4.5.1.1). The time series in this dataset set of human activities are labeled, so we can use the class information to make some conclusions on the subsequences by our search procedure. For this purpose, we compared the class of each query to the subsequence found when using $w = 0.1$ and $w = 0.5$. In most cases, either both parameter values would make the correct classification, or both would make a mistake in a 1-NN classification. When the decisions are opposite to each other, the classification using $w = 0.5$ is correct. Specifically, it happens in four cases – in the total of 20 combinations of query and number of observations.

The UCR Time Series repository is the largest repository of time series datasets for clustering and classification. The accuracy rates obtained by the nearest neighbor algorithm using DTW in each dataset are presented in the repository's website in two different ways: (i) using no warping window, i.e., the relative length of the window is 100% and (ii) warping-constrained DTW, in which the best value for the warping window length is learned in the training data. In some datasets, the warping window taken as optimal is larger than 10% (around 15% of the datasets). Even more, in several cases, the performance of DTW with no warping window is better than its constrained version.

Besides the recommendations for the 1-NN classification, there are no conclusive studies about this parameter for different algorithms or mining tasks. [Silva and Batista \(2016b\)](#) performed an experiment on hierarchical clustering varying the warping window size in several datasets. The results show that more than a half of the best results were obtained by using warping windows larger than 10% of the time series' length, being several of them obtained using no warping window at all.

Figure 47 shows a small and intuitive example of data in which a large warping window is necessary. The time series used in this example are from the dataset REFIT (c.f. Section 4.5.1.6). It is evident, by visual inspection, that a similar cycle of the dishwasher generated two of the presented time series, but with different durations (e.g. the user chose a longer washing cycle, more suitable for greasy dishes). Nevertheless, when we cluster the time series by the single-linkage hierarchical clustering using DTW with 10% of relative warping window, the best match in this simple subset is wrong. We can obtain a similar clustering using 20% of warping, with changes only in the scale of the distances. When we use a larger warping window, however, the time series are correctly clustered.

Despite that we used a hierarchical clustering algorithm to exemplify the need for large warping window, the result presented in Figure 47 has a direct impact on the similarity search. Consider that the objects at the bottom and the top of the figure are part of the reference time series. If we search the nearest neighbor of the object in the middle using a small window, the search will return the object at the bottom instead of the correct one (at the top).

Figure 47 – Single-linkage clustering obtained by using DTW with relative warping window lengths of 10% (*left*) and 50% (*right*). Note that the correct cluster is only obtained by using the larger warping constraint

Source: Elaborated by the author.

In this dataset, the UCR-USP suite is approximately 3 times faster than the UCR suite with a warping window of 50% and a query with 1,024 observations (approximately 2 hours and 15 minutes of monitoring, c.f. Section 4.6.1).

4.7 Pruning Paths on DTW Variations and Other Distance Measures

We have stated so far the relevance of DTW on the similarity search. In fact, it is arguably the most used distance measure in time series similarity search. However, one may be interested in using another distance measure for the nonlinear alignment between time series.

In the last decades, DTW was modified in several different ways, to provide robustness to certain variances found in specific application domains. Some examples are the Derivative DTW (KEOGH; PAZZANI, 2001), the Weighted DTW (JEONG; JEONG; OMITAOMU, 2011), and the Prefix and Suffix Invariant DTW (SILVA; BATISTA; KEOGH, 2016). Also, some distance measures find a nonlinear alignment while respect the triangle inequality, i.e., they are metrics calculated by a dynamic programming algorithm similar to DTW. Some examples are the Time Warp Edit Distance (MARTEAU, 2009) and the Move-Split-Merge (STEFAN; ATHITSOS; DAS, 2013).

Speeding up the similarity search under different nonlinear alignment distance measures is a widely studied topic in the time series literature (WANG *et al.*, 2013). In these cases, the procedure to avoid distance calculations may vary, depending on the distance measure. For instance, when using the distance metrics, we can apply indexing structures based on the triangle inequality to reduce the search space (HJALTASON; SAMET, 2003).

On the other hand, we are not aware of strategies similar to the PrunedDTW. We believe that adapting PrunedDTW to other distance measures can have a similar effect to the

ones demonstrated in this work: improve the worst case of the similarity search, tackling the bottleneck of the search procedure.

Most of the distance measures for time series comparison are based on minimizing the cost of matching the observations. In these cases, the steps of PrunedDTW are fully repeated to prune unpromising alignment paths. However, the UB needs to be calculated accordingly. In general, the *bsf* can be used, as described in Section 4.4.

It is important to notice that some algorithms to compare time series are based on maximizing an objective function. One example of these methods is the Longest Common Subsequence (LCSS). While DTW and its variations aim to minimize the total cost of matching the observations, LCSS maximizes the length of the similar subsequence between the time series under comparison. In this case, the roles of LB and UB are inverted. To prune the LCSS calculations between pairs of subsequences, we first need to calculate a UB (VLACHOS *et al.*, 2006). Conversely, to prune unpromising partial alignments, we need to replace the UB used by PrunedDTW by an LB function. As a consequence, the comparisons used in the pruning decision step also need to be inverted. Specifically, we need to invert the operators $<$ and $>$, as well as $\leq$ and $\geq$.

Adapting the pruning of unpromising alignment paths for these distance measures can be subtle, and we let a deeper discussion and experimentation as part of the future work. The next section concludes this work and briefly introduces other ideas to develop in the future.

4.8 Conclusion

In this work, we embedded a recent advance on speeding up Dynamic Time Warping into the similarity search scenario. This approach is motivated by the fact that this algorithm can speed up the distance calculations in the cases in which the current similarity search methods perform worst. Specifically, we identified that the DTW calculations configure the bottleneck of the subsequence similarity search.

We have shown that our method can speed up the fastest tool for similarity search under DTW. Also, our method achieves the highest speed up rates for long queries and large warping windows, the worst case for the usual indexing techniques. When the queries and the warping windows are small, our method achieves similar runtime when compared to the state-of-the-art.

We notice that embedding the PrunedDTW in the similarity search procedure is not necessarily the ultimate solution to mitigate the observed bottleneck for all scenarios. For instance, if we need to perform the similarity search under sparse time series data, we may use a method specific for this case (MUEEN *et al.*, 2016). If an approximate solution is considered suitable for the problem, an algorithm that approximates DTW – such as FastDTW (SALVADOR;

[CHAN, 2007](#)) – can be applied. However, PrunedDTW is exact and can be applied in any case, improving the efficiency as demonstrated by our experimental evaluation.

As a practical overview of our contribution, consider that one gets an average speedup ratio of 2, i.e., the UCR-USP Suite spends half of the time then the UCR Suite to search a set of queries. This speedup is a common achievement of our method (see, for instance, Figure 44). It means that a laboratory of medical analysis can serve twice the number of patients per day, for example. Similarly, an industry needs to spend half the computational power for monitoring cauldrons and machines. After all, the speedup achieved by the UCR-USP Suite is directly proportional to the savings, profit or any other benefits obtained by applying it.

As future work, we intend to evaluate the use of the adapted PrunedDTW in tasks which use the nearest neighbor search as an intermediate step. Some examples are the classification ([DING *et al.*, 2008](#)) and clustering ([BEGUM *et al.*, 2015](#)) of time series. Also, we intend to extend the proposed method to multidimensional time series ([SHOKOOHI-YEKTA *et al.*, 2017](#)).

FAST SIMILARITY MATRIX PROFILE FOR MUSIC ANALYSIS AND EXPLORATION

Abstract: Most algorithms for music data mining and retrieval are based on analyzing the similarity between feature sets extracted from the raw audio. A common approach to assess similarities within or between recordings is to create similarity matrices. However, this approach requires quadratic space for each comparison and typically requires a costly post-processing of the matrix. In this work, we develop a simple and efficient representation that is based on a subsequence similarity join, which may be used in several music analysis tasks. In addition, we introduce a technique to drastically speed up its calculation. We demonstrate how the proposed representation can be exploited for multiple applications, focusing on the cover song recognition problem.

5.1 Introduction

With the growing interest in applications related to music processing, music information retrieval and data mining has vastly attracted attention in both academia and music industry. However, the analysis of audio recordings remains a significant challenge, aggravated by the increasing volume of music data caused by the expansion of the electronic music files distribution and streaming services. In this scenario, algorithms for music analysis must be efficient in both time and space.

Most algorithms for content-based music analysis have at their cores some similarity or distance function. Consequently, a variety of applications rely on some technique to assess the similarity between music objects. These applications include: segmentation ([SERRA *et al.*, 2014](#)), audio-to-score alignment ([CARABIAS-ORTI *et al.*, 2015](#)), cover song recognition ([SERRA *et al.*, 2008](#)), query by singing/humming ([LIU, 2014](#)), and visualization ([WU; BELLO, 2010](#)).

A common approach to assessing similarity in music recordings is utilizing a self-similarity matrix (SSM) (FOOTE, 1999). This representation reveals the relationship between each snippet of a track to all the other segments in the same recording. This idea has been generalized to measure the relationships between subsequences of different songs, such as the application of cross-recurrence analysis for cover song recognition (SERRA; SERRA; ANDRZEJAK, 2009).

Because similarity matrices simultaneously reveal both the global and local structure of music recordings, they are highly advantageous. However, this representation requires quadratic space in relation to the length of the feature vector used to describe the audio. For this reason, most methods used to find patterns in the similarity matrix are (at least) quadratic in time complexity. However, most information contained in similarity matrices is irrelevant or has little impact on its analysis. This observation suggests the need for a more space and time efficient representation of music recordings.

In this work, we extend the all-pairs-similarity-search of subsequences, which is also known as similarity join, in order to assess the similarity between audio recordings for MIR tasks. Analogously to the similarity matrices, representing the entire subsequence join requires a quadratic space, and it also has a high time complexity, which is dependent upon the length of the subsequences to be joined.

We demonstrate how to exploit a new data structure called a matrix profile, which allows for a space efficient representation of the similarity join matrix between subsequences. Moreover, we can leverage recent optimizations in the all-neighbor search that allow the matrix profile to be computed efficiently (MUEEN *et al.*, 2017). For clarity, we refer to the representation presented in this paper as Similarity Matrix Profile (SiMPle).

Figure 48 illustrates an example of two matrices representing the dissimilarities/distances within and between recordings and their relative SiMPle, which corresponds to the minimum value of each column in the matrices.

In summary, our method has the following advantages/features:

- It is a novel approach to assess the audio similarity, and it can be used in several MIR algorithms;
- We exploit the fastest known subsequence similarity join technique in the literature, which makes our method fast and precise;
- It is simple and only requires a single parameter, which is intuitive to set for MIR applications;
- It is space efficient, requiring the storage of only $\mathcal{O}(n)$ values;

Figure 48 – Self-distance matrix (*left*), the cross-distance matrix between different recordings (*right*), and their respective SiMPle (*bottom*)

Source: Elaborated by the author.

- Once we calculate the similarity profile for a dataset, it can be efficiently updated, which has important implications for streaming audio processing.

The remainder of the paper is organized as follows. Section 5.2 introduces the SiMPle and our novel algorithm to speed up its calculation. Section 5.3 presents a cover song recognition system based on SiMPle and how it can be extended to identify covers in a streaming fashion. Next, Section 5.4 presents an experiment on the scalability of the proposed method in that task. Section 5.5 augments the applicability of SiMPle for different music analysis tasks. Finally, Section 5.6 concludes this work.

5.2 SiMPle: Similarity Matrix Profile

We begin this section by presenting the necessary definitions to introduce our method. We will use the terms time series and subsequences in reference to the feature vectors that describe both the whole audio and excerpts, respectively. Formally, a time series is defined as follow.

Definition A time series $T = (t_1, t_2, \dots, t_n)$ is a contiguous sequence of vectors t_i with length n , such that t_i is composed of f real values comprising the features extracted to represent a small segment of the audio.

For clarity, if we are using chromatic features to describe our audio files, f is the number of bins adopted (usually 12, 24, or 32) and n is the number of windows from which the features were extracted. Next, we define a subsequence.

Definition A subsequence $S = (t_s, t_{s+1}, \dots, t_{s+m-1})$ is a contiguous subset from the time series T with length m .

The main operation for producing the similarity matrix profile is the similarity join, which will be defined below.

Definition Similarity join. Given two time series A and B and the desired subsequence length m , the similarity join identifies the nearest neighbor of each subsequence in A from all the possible subsequence set of B (both with length m).

Through a similarity join, we can gather two pieces of information about each subsequence in A , which are *i*) the Euclidean distance to its nearest neighbor in B and *ii*) the position of its nearest neighbor in B . This information can be compactly stored in vectors, referred to as a similarity matrix profile (SiMPle) and similarity matrix profile index (SiMPle index), respectively.

When both input time series refer to the same recording, this is a special case of similarity join. We define the operation that handles this specific case as a self-similarity join.

Definition Self-similarity join. Given a time series A and the desired subsequence length m , the self-similarity join identifies the nearest neighbor of each subsequence from A to every (non-trivial) subsequence set from A .

The only major difference between the self-similarity join (Definition 5.2) and the AB-similarity join (Definition 5.2) is the exclusion of trivial matched pairs when identifying the nearest neighbor. The exclusion of trivial matches is crucial, since matching a subsequence with itself (or slightly shifted version of itself) produces no useful information.

We describe the original method to calculate SiMPle (SILVA *et al.*, 2016) in the Algorithm 7. Despite proposing a new method, defining the original algorithm is important to better understand our proposal and for comparison purposes. In line 1, we record the length of B . In line 2, we allocate memory and initialize SiMPle P_{AB} and SiMPle index I_{AB} . From line 3 to line 6, we calculate the distance profile vector D , which contains the distances between a given subsequence in time series B and each subsequence in time series A . The particular function we used to compute D is MASS, which is the most efficient algorithm known for distance vector computation (MUEEN *et al.*, 2017). Then we perform the computation of pairwise minimum for each element in D with the paired element in P_{AB} (i.e., $\min(D[i], P_{AB}[i])$ for all $i = 0$ to $\text{length}(D) - 1$.) We also update $I_{AB}[i]$ with idx , when $D[i] \leq P_{AB}[i]$ as we perform the pairwise minimum operation. Finally, we return P_{AB} and I_{AB} in line 7.

Algorithm 7 – Procedure to calculate SiMPle and SiMPle index**Requer:** Two time series, A and B , and the desired subsequence length m **Assegure:** The SiMPle P_{AB} and the associated SiMPle index I_{AB}

```

1:  $n_a \leftarrow \text{length}(A)$ 
2:  $P_{AB} \leftarrow \text{infs}(n_A - m + 1), I_{AB} \leftarrow \text{zeros}(n_A - m + 1), \text{idxes} \leftarrow 1 : n_A - m + 1$ 
3: para todo  $\text{idx}$  in  $\text{idxes}$  faça
4:    $D \leftarrow \text{MASS}(A[\text{idx} : \text{idx} + m - 1], B)$  ▷ c.f. (MUEEN et al., 2017)
5:    $P_{AB}, I_{AB} \leftarrow \text{ElementWiseMin}(P_{AB}, I_{AB}, D, \text{idx})$ 
6: fim para
   retorna  $P_{AB}, I_{AB}$ 

```

Note that the Algorithm 7 computes SiMPle for the general similarity join. To modify it to compute the self-similarity join SiMPle of a time series A , we simply replace B by A in line 4 and ignore trivial matches in D when performing *ElementWiseMin* in line 5.

The method MASS (used in line 4) is important to speed up the similarity calculations. The main contribution of MASS is the use of an efficient Fast Fourier Transform-based method to calculate the cross-correlation between a subsequence and a time series. The cross-correlation provides the sliding dot product between the elements of both sequences. In other words, it provides the dot products between the subsequences under analysis and each subsequence of the same length from the other time series – or the same time series, in the case of self-join. Figure 49 illustrates this idea.

Figure 49 – The cross-correlation between a time series B and a subsequence from A provides the dot product between their elements

Source: Elaborated by the author.

This trick can be used with several distance measures which depend on the dot product between values. In this work, we use the squared Euclidean distance. In order to clarify the use of the cross-correlation, Equation 5.1 defines the Euclidean distance, where S_A and S_B are subsequences from A and B , respectively.

$$\begin{aligned}
 ED(S_A, S_B) &= \sum_{i=1}^m (a_i - b_i)^2 \\
 &= \sum_{i=1}^m (a_i)^2 + \sum_{i=1}^m (b_i)^2 - 2(S_A \cdot S_B)
 \end{aligned} \tag{5.1}$$

For each subsequence in A , we create a distance profile, which represents the distance between the currently assessed subsequence to every subsequence of the same length in B . Once we have calculated the sliding dot product for this subsequence, the distance profile is directly obtained by applying Equation 5.1 to each position.

The original algorithm to calculate SiMPle (inspired by STAMP (YEH *et al.*, 2016)) uses MASS for each subsequence. Recently, researchers noticed that the dot products do not need to be recalculated from the scratch for each subsequence (ZHU *et al.*, 2016). Instead, we can reuse the values calculated for the first subsequence to make a faster calculation in the next iterations. The idea is to make use of the intersections between the required products in consecutive iterations. Consider the subsequence $(a_i, a_{i+1}, \dots, a_{i+m-1})$ of length m extracted from the time series A , starting at the position i . After the calculation of the distance profile to this subsequence, we have all the dot products $(a_i, a_{i+1}, \dots, a_{i+m-1}) \cdot (b_i, b_{i+1}, \dots, b_{i+m-1})$. By subtracting the first partial of these products, i.e., the value a_i multiplied by each value in the time series B , we obtain the products given by $(a_{i+1}, a_{i+2}, \dots, a_{i+m-1}) \cdot (b_{i+1}, b_{i+2}, \dots, b_{i+m-1})$. Finally, we can obtain $(a_{i+1}, a_{i+2}, \dots, a_{i+m}) \cdot (b_{i+1}, b_{i+2}, \dots, b_{i+m})$ by adding $(a_{i+m})(b_{j+m})$ for each j . The last dot product gives the required value to the next iteration. Figure 50 illustrates the procedure.

Figure 50 – In order to quickly obtain the sliding dot product between a subsequence from the time series A and the time series B , our method reuses the products obtained from the previous subsequence in A

Source: Elaborated by the author.

After this procedure, we have all the values necessary to construct the distance profile that refers to the subsequence $(a_{i+1}, a_{i+2}, \dots, a_{i+m})$, except for the first value. For this reason, before calculating the SiMPle, we calculate the cross-correlation of the first subsequence from B , i.e., $(b_1, b_2, \dots, b_m)$ with the time series A . With this procedure, we obtain every necessary dot product to fulfill the distance profile calculations. Algorithm 8 presents this procedure in detail, which we will refer in this paper as SiMPle-Fast.

5.3 SiMPle-Based Cover Song Recognition

“Cover song” is the generic term used to denote a new performance of a previously recorded track. For example, a cover song may refer to a live performance, a remix, or an interpretation in a different music style. The automatic identification of covers has several applications, such as copyright management, collection organization, and search by content.

Algorithm 8 – SiMPle-Fast**Requer:** Two time series, A and B , and the desired subsequence length m **Assegure:** The SiMPle P_{AB} and the associated SiMPle index I_{AB}

```

1:  $firstprod, sumx2 \leftarrow MASS(A[1 : m], B)$ 
2:  $currprod, sumy2 \leftarrow MASS(B[1 : m], A)$ 
3:  $dropval \leftarrow A[1]$ 
4:  $n_p \leftarrow length(currprod)$ 
5:  $n_a \leftarrow length(A)$ 
6:  $P_{AB} \leftarrow \text{infs}(n_A - m + 1), I_{AB} \leftarrow \text{zeros}(n_A - m + 1), \text{idxes} \leftarrow 2 : n_A - m + 1$ 
7: para todo  $idx$  in  $\text{idxes}$  faça
8:    $subseq \leftarrow A[idx : idx + m - 1]$ 
9:    $sumy2 \leftarrow sumy2 - dropval^2 + subseq(m)^2$ 
10:   $currprod[2 : n_p] \leftarrow currprod[1 : n_p - 1] + (subseq[m] \cdot B[m + 1 : m + n_p]) - (dropval \cdot B[1 : n_p - 1])$ 
11:   $currprod[1] \leftarrow firstprod[i]$ 
12:   $dropval \leftarrow subseq[1]$ 
13:   $distProf \leftarrow \text{sum}_{all\_dimensions}(sumx2 - 2 \cdot currprod + sumy2)$ 
14:   $P_{AB}, I_{AB} \leftarrow \text{ElementWiseMin}(P_{AB}, I_{AB}, distProf, idx)$ 
15: fim para
    retorna  $P_{AB}, I_{AB}$ 

```

In order to identify different versions of the same song, most algorithms search for globally (TSAI; YU; WANG, 2008) or locally (SERRA *et al.*, 2008; SILVA; SOUZA; BATISTA, 2015) conserved structure(s). A well-known and widely applied algorithm for measuring the global similarity between tracks is Dynamic Time Warping (DTW) (MÜLLER, 2007). Despite its utility in other domains, DTW is not generally robust enough to handle the differences in structure between the recordings. A potential solution would be segmenting the song before applying the DTW similarity estimation. However, audio segmentation itself is also an open problem, and the error on boundary detection can cause a domino effect (compounded errors) in the whole identification process.

In addition, the complexity of the algorithm to calculate DTW is $\mathcal{O}(n^2)$. Although methods to approximate the DTW with fast algorithms have been proposed (SALVADOR; CHAN, 2007), there is no error bound for such approximations. In other words, it is not possible to set a maximum error in the value obtained by it in relation to the actual DTW.

Algorithms that search for local similarities have been successfully used to provide structural invariance to the cover song identification task. A well-known method for music similarity proposes the use of a binary distance function to compare chroma-based features, followed by a dynamic programming local alignment (SERRA *et al.*, 2008). Despite its demonstrated utility to recognize cover recordings and some variants proposed in the literature, this method has several parameters that are unintuitive to tune, and is also slow. Specifically, the local alignment is estimated by an algorithm with similar complexity to DTW. Plus, the binary distance between chroma features used in each step of the algorithm relies on multiple shifts of

the chroma vectors under comparison. In this work, we propose to use SiMPle to measure the distance between recordings in order to identify cover songs. In essence, we exploit the fact that the global relation between the tracks is composed of many local similarities. In this way, we are able to simultaneously take advantage of both local and global pattern matching.

Intuitively, we should expect that the SiMPle obtained by comparing a cover song to its original version is composed mostly of low values. In contrast, two completely different songs will result in a SiMPle constituted mainly by high values. For this reason, we adopted the median value of the SiMPle as a global distance estimation. Formally, the distance between a query B and a candidate original recording A is defined in Equation 5.2.

$$d(A, B) = \text{median}(\text{SiMPle}(B, A)); \quad (5.2)$$

Note that several other statistical measures could be used instead of the median. However, the median is robust enough to handle outliers in the matrix profile. Distortions may appear when a performer decides, for instance, to add a new segment (e.g., an improvisation or drum solo) to the song. The robustness of our method in this situation, as well as other changes in structure, is discussed in the next section.

5.3.1 On the Structural Invariance

The structural variance is a critical concern when comparing different songs. Changes in structure may occur by insertion or deletion of segments, as well as changes in the order that different excerpts are played. From a high-level point of view, SiMPle describes a global similarity outline between songs by providing information about local comparisons. This fact has several implications in our distance estimation, which makes it largely invariant to structural variations:

- If two performances are virtually identical, except for the order and the number of repetitions of each representative excerpt (i.e., chorus, verse, bridge, etc.), all the values that compose SiMPle are close to zero.
- If a segment of the original version is deleted in the cover song, this will cause virtually no changes in the SiMPle.
- If a new feature is inserted into a cover, the only consequence is a peak in the SiMPle, which may only slightly increase its median value.

5.3.2 Experimental Evaluation

The evaluation of different choices of features sets is not the main focus of this paper. For this reason, we fix the use of chroma-based features in our experiments, because it is the

most popular feature set in music similarity retrieval and cover song identification, scenarios in which timbral and rhythmic features usually fail to achieve good results (FU *et al.*, 2011).

In order to provide local tempo invariance, we used the chroma energy normalized statistics (CENS) (MÜLLER; KURTH; CLAUSEN, 2005). Specifically, for the cover song recognition task, we adopted the rate of two CENS per second of audio.

In addition, we preprocessed the feature sets in each comparison to provide key invariance. Before calculating the similarity between songs, we transposed one of them in order to have the same key using the optimal transposition index (OTI) (SERRA *et al.*, 2008). Note that chroma features are not necessarily the best option for all the experiments presented in this paper. Chroma features were chosen because of its flexibility, the vast amount of work using it, and the availability of tools to extract these kind of features from the audio. In recent years, the community has witnessed advances in feature learning for different music applications (OORD; DIELEMAN; SCHRAUWEN, 2013; SIGTIA; DIXON, 2014), including learning chromatic features (KORZENIOWSKI; WIDMER, 2016; FANG; DAY; CHANG, 2016). However, we emphasize that any feature extraction approach that provides the temporal variation of the evaluated recordings can be used with our approach.

5.3.2.1 Datasets

We evaluate our method in different scenarios regarding music styles and the size of the databases. Specifically, we tested the proposed distance measure’s utility for assessing both popular and classical recordings. The first database considered is the YouTube Covers (SILVA; SOUZA; BATISTA, 2015), which consists in 50 different compositions, each containing seven different recordings obtained from YouTube videos. The data was originally split into training and testing partitions, in which the training set is composed of the original studio recording and a live version performed by the same artist. To allow comparisons with the literature, we follow the same configuration.

The second dataset we consider is the widely used collection of Chopin’s Mazurkas (SAPP, 2017). The set of Mazurkas used in this work contains 2,919 recordings of 49 pieces for piano. The number of recordings of each song varies from 41 to 95.

5.3.2.2 Results and Discussion

In order to assess the performance of our method, we used three commonly applied evaluation measures: mean average precision (MAP), precision at 10 (P@10), and the mean rank of first correctly identified cover (MR1). Note that for MR1, smaller values are better. For both the YouTube Covers and Mazurkas datasets, we compared our algorithm by using results previously presented in the literature. In addition to comparing the results presented in the paper for which the dataset was created (SILVA; SOUZA; BATISTA, 2015), in the former case, we carefully implemented the algorithm for local alignments based on the chroma binary distance (SERRA *et*

al., 2008). Table 4 shows the results. Given this dataset only has two recordings per song in the training set, notice that the maximum value for P@10 is 0.2.

Table 4 – Mean average precision (MAP), precision at 10 (P@10), and mean rank of first correctly identified cover (MR1) on the YoutubeCovers dataset

Algorithm	MAP	P@10	MR1
DTW	0.425	0.114	11.69
Silva et al. (SILVA; SOUZA; BATISTA, 2015)	0.478	0.126	8.49
Serrà et al. (SERRA <i>et al.</i> , 2008)	0.525	0.132	9.43
SiMPle	0.591	0.140	7.91

Our method achieved the most accurate results in this experiment. In addition, our method is notably faster than the second best (Serrà et al.). For a better understanding of the runtimes, we tested all the algorithms used in this experiment. While the original algorithm to calculate SiMPle is competitive with DTW and our new method can perform ten times faster (c.f. Section 5.4 for details), the method proposed by Serrà et al. is more than ten times slower than DTW in the best case. We acknowledge that we did not prioritize optimizing the competing method. However, we do not believe that any code optimization is capable of significantly reducing the performance gap.

We also consider the Mazurkas dataset. In addition to the results achieved by DTW, we report MAP results documented in the literature, which were achieved by retrieving the recordings by structural similarity strategies using this data. Specifically, the subset of mazurkas used in this work is exactly the same as those used in (BELLO, 2011) and (SILVA *et al.*, 2013) and has only minor differences from the dataset used in (GROSCHÉ *et al.*, 2012). Although variations of (SERRA *et al.*, 2008) are commonly used in state-of-the-art cover song identification systems (CHEN; LI; XIAO, 2017), we do not include its results due to the high time complexity. Table 5 shows the results.

Table 5 – Mean average precision (MAP), precision at 10 (P@10), and mean rank of first correctly identified cover (MR1) on the Mazurkas dataset

Algorithm	MAP	P@10	MR1
DTW	0.882	0.949	4.05
Bello (BELLO, 2011)	0.767	-	-
Silva et al. (SILVA <i>et al.</i> , 2013)	0.795	-	-
Grosche et al. (GROSCHÉ <i>et al.</i> , 2012)	0.525	-	-
SiMPle	0.880	0.952	2.33

The structures of the pieces on this dataset are respected in most of the recordings. In this case, DTW performs similarly than our algorithm. However, our method is approximately one order of magnitude faster, and it has several advantages over DTW such as an increased incremental property, which is discussed in the next section.

5.3.2.3 Streaming Cover Song Recognition

Real-time audio matching has attracted attention in the last years. In this scenario, the input is a stream of audio and the output is a sorted list of similar objects in a database.

In this section, we evaluate our algorithm in an online cover song recognition scenario. For concreteness, consider a TV station broadcasting a live concert. In order to automatically present the name of the song to the viewers or to synchronize the concert with a second screen app, we would like to take the streaming audio as input for our algorithm and recognize what song the band is playing as soon as possible. To accomplish this task, we need to match the input to a set of (previously processed) recordings.

In addition to allowing the fast calculation of all the distances of a subsequence to a whole song, the proposed algorithm has an incremental property that can be exploited to estimate cover song similarity in a streaming fashion. If we have a previously calculated SiMPle, then, when we extract a new vector of (chroma) features, we do not need to recalculate the whole SiMPle from the beginning. Instead, only two quick steps are required:

- First it is necessary to calculate the distance profile to the new subsequence, i.e., the distance of the last observed subsequence (including the new feature vector) to all the subsequences of the original song.
- Then it is necessary to update SiMPle by selecting the minimum value between the new distance profile and the previous SiMPle for each subsequence.

Notice that in the first step, we do not need to calculate the distance profile from the scratch. Instead, we take advantage of the previously calculated values (c.f. Section 5.2). For this reason, we also need to update these values after the first step.

To evaluate the capability of our method for streaming recognition, we performed a simple experiment simulating the previously described scenario. First, we extracted features from each track in the dataset of original recordings. For clarity, we will refer to this database as the training set. Then, we randomly chose another recording as our query and processed it in the following manner. We began extracting features from the first three seconds of the query to calculate the first distance estimation to each training object. After this initial step, for each second of the query, we repeated the process of extracting features and re-estimating the distance measure to the training set.

In this experiment, we used the Mazurkas dataset with two CENS per second. The training set is composed of one recording of each piece. We used a performance (which is not part of the training set) with approximately 92 seconds as a query, and the process was still faster than real-time. Specifically, the updates took approximately 0.28 second to extract the features, update SiMPle, and recalculate the distance to all the training objects for each second of “listened” performance.

Despite previously reporting a real-time matching of streaming audio by using the original algorithm to calculate the SiMPle (SILVA *et al.*, 2016), it was not possible in the new experiment. For the 92 seconds audio, the whole procedure took 112 seconds. In other words, there was little delay in the processing.

Figure 51 demonstrates the changes in distance estimation in an audio streaming query session. In this case, we used a recording of the “Mazurka in F major, Op. 68, No. 3” as the query and a subsequence of two seconds. In the first distances estimation, i.e., after two seconds of audio, the correct class appears in the sixth position in the ranking. However, with more evidence, it quickly becomes the best match. Specifically, after three seconds of streaming, it goes to the fourth position. Four seconds from the beginning, the correct class is already considered the best match. It stands until the streaming ends.

Figure 51 – Changes in the distance when querying a recording of the “Mazurka in F major, Op. 68, No. 3” in a streaming fashion. The graphs represent the top five matches after processing two (left), three (middle), and four (right) seconds of the audio

Source: Elaborated by the author.

5.4 Scalability

In the previous section, we briefly presented results on the runtime of the experimented methods. In this section, we present a formal experiment to gain a better estimation of the differences in the time performance of the methods.

Both the length of the time series and the number of tracks in the database directly affect the runtime of our method. For this reason, we experimented both parameters in separate experiments.

In our first experiment, we randomly chose one recording on the YouTube Covers dataset as a query and measured the similarity to all other tracks in the dataset. After each distance calculation, we stored the cumulative runtime of the experiment. This experiment shows how the runtime increases according to the size of the dataset, i.e., the number of tracks to be compared to the query. Additionally, it has been made for different feature rates, which provides us with different time series lengths. Figure 52 presents the results.

Figure 52 – Runtime obtained by querying one song when varying the number of objects in the dataset and the length of the time series. Specifically, we used 5 (*left*) and 100 (*right*) windows per second in the feature extraction process

Source: Elaborated by the author.

The results show a linear behavior of the three methods regarding the size of the dataset. In any of the cases, the proposed method is clearly faster than the others. Specifically, for the case of 100 features per second (around 1,600 features per song in average after the smoothing provided by CENS), the SiMPle-Fast is 8.5 times faster than DTW and 6.6 times faster than the original SiMPle implementation. In the case of using 5 windows per second, i.e., 132 CENS features per song in average, our method is approximately 2.7 times faster than DTW and 3.3 times faster than SiMPle.

Another interesting aspect to observe is the relation between the runtime of SiMPle and DTW. The method that achieves the best runtime between them is dependent on the size of the time series. For this reason, we realized the necessity for a second experiment, in which we evaluate the influence of the time series length on the runtime. For this, we used six different feature rates, providing us the runtime for eight different lengths of time series. For better precision, each experiment was executed five times and the presented points were obtained by the average of the runtime values for each time series length. Figure 53 presents the results.

Figure 53 – Average runtime for querying five random queries to the remaining examples on the YouTube Covers dataset by varying the length of the time series, given by the number of features per second. The markers represent actual runtime measurements, and the lines were obtained by a quadratic interpolation curve fitting

Source: Elaborated by the author.

In the second experiment, there is a quadratic trend on the runtime with respect to the increasing length of the time series. Despite the fact that the original SiMPle and the DTW have a similar performance in this case, the SiMPle-Fast is notably faster than both. More importantly, these results show the high scalability of our method; the longer the time series, the greater the improvement provided by our method.

5.5 Music Data Exploration using SiMPle

In this work, we focus on assessing music similarity by joining subsequences. While we evaluate our method on the cover song recognition task, we claim that the SiMPle is a powerful tool for other music processing tasks. To reinforce this argument, we present insights on using SiMPle in different application domains, as well as some initial results. The methods presented in this section may have room for improvement, but they are simple yet effective. We intend to further explore and evaluate SiMPle in (at least) the tasks listed below.

In contrast to the previous experiments, when we use the self-similarity join to highlight points of interest in a recording, we apply ten CENS per second.

5.5.1 Discord and Repeated Patterns

The SiMPle from a self-similarity join has several exploitable properties. For example, the lowest points correspond to the locations of the most faithfully repeated section (i.e., the chorus or refrain). In the time series literature, these subsequences are referred to a *best motif pair*, a term which we will use in this section. Once the best motif pair is found, other definitions of motifs can be used with minor additional calculations (MUEEN, 2014). For instance, in order to use SiMPle to find melodic motifs – repeated several times during the song – one may simply annotate the subsequences corresponding to values smaller than a threshold in each distance profile. Then, it is possible to analyze the patterns according to location and frequency.

On the other hand, the highest point on the SiMPle corresponds to the “most unique” snippet from the recording. Searching for a subsequence that is farthest from any other, which is known as discord discovery, can be used in music processing to find interesting segments in recordings. It can be used to identify a solo, improvisational segments, or the bridge.

For example, consider the song “Let It Be” by The Beatles. Figure 54 shows the SiMPle obtained for this track and points to their discord and faithfully repeated sections.

While the best motif pair point to refrains, the discord includes bridge and the beginning of the guitar solo.

Figure 54 – The most reliable repeated pair in a recording is determined by the subsequences starting at the positions of the minimum values of its SiMPle. At the same time, the position of the highest value points to the beginning of the discord excerpt

Source: Elaborated by the author.

5.5.2 Audio Thumbnailing

Audio thumbnails are short representative excerpts of audio recordings. Thumbnails have several applications in music information retrieval. For example, they can be used to show the result of a search to the user. In a commercial application, they can be used as the preview to a potential customer in an online music store.

There is a consensus in the MIR community that the “ideal” music thumbnail is the most repeated excerpt, such as the chorus (BARTSCH; WAKEFIELD, 2005). By using this assumption, the application of SiMPle to identify a thumbnail is direct. Consider the SiMPle index obtained by the self-join procedure. The thumbnail is given by the subsequence starting in the position that is most used as the nearest neighbor. In other words, the beginning of the thumbnail is given by the position related to the mode of the SiMPle index.

To illustrate this idea, we considered the song “New York, New York” by Frank Sinatra. Looking for a 30 seconds thumbnail, we found an excerpt that is comprised of the last refrain, as well as the famous (brass) instrumental basis of the song. Figure 55 shows the histogram of the SiMPle index found in this experiment.

Figure 55 – Histogram of the SiMPle index for the song “New York, New York.” Each bar counts how many times the subsequence started at that point was considered the nearest neighbor of any other. We consider the subsequence represented by the most prominent peak as the thumbnail for this recording

Source: Elaborated by the author.

5.5.3 Visualization

Visualization tools for different music aspects are an important instrument in the comprehension of the music content (CANTAREIRA; NONATO; PAULOVICH, 2016). Introduced in (WATTENBERG, 2002), the arc diagram is a powerful tool to visualize repetitive segments in MIDI files (WATTENBERG,) and audio recordings (WU; BELLO, 2010). This approach represents a song by plotting arcs linking repeated segments, enhancing the understanding of structural elements.

All the information required to create such arcs is completely comprised on the SiMPle and the SiMPle index, which is obtained by a self-join. Specifically, SiMPle provides the distances between subsequences, which can be used to determine if they are similar enough to put a link between them and to define the color or transparency of each arc. The SiMPle index can be used to define both the positions and width of the arcs.

Figure 56 shows the scatter plot of the SiMPle index for “Hotel California” by Eagles. In this figure, there is a point (x, y) only if y is the nearest neighbor of x . The clear diagonals on this plot represent regions of n points such that the nearest neighbors of $[x, x + 1, \dots, x + n - 1]$ are approximately $[y, y + 1, \dots, y + n - 1]$. If the distance between such excerpts is low, then these regions may have a link between them. For this example, we defined the mean value of the SiMPle in that region as the distance threshold between the segments, in order to decide if they should establish a link. This threshold has a direct impact on the number of arcs plotted.

Figure 56 – Scatter plot of the SiMPle index for the song “Hotel California.” The (light) gray area indicates a possible link, but only the values in the (dark) green area represent subsequences with distance lower than the threshold

Source: Elaborated by the author.

By using a straightforward algorithm to find such diagonals, we only need to define a threshold of distance and minimum length of the linkages. We set the width of the links in our experiment to be greater than or equal to five seconds. Figure 57 shows the resulting arc plot for the example shown in Figure 56.

Figure 57 – Arc plot for the song “Hotel California.” These plots show the difference between using the mean value of SiMPle as a distance threshold (*above*) and no distance threshold at all (*below*). The color of the arcs are related to their relevance, i.e., as darker the arc, closer the subsequences linked by it

Source: Elaborated by the author.

5.5.4 Endless Reproduction

Consider a music excerpt s_1 , which starts at the time t_1 of a specific song, has a small distance to its nearest neighbor s_2 , which starts at time t_2 . When the reproduction of this song arrives t_1 , we can make a random decision to “skip” the reproduction to t_2 . Given that s_1 and s_2 are similar, this jump may be imperceptible to the listener. By creating several points of skip, we are able to define a sequence of jumps that creates an endless reproduction of the song. A well-known deployed example of this kind of player is the Infinite Jukebox ([LAMERE,](#)).

The distance values obtained by the self-join represent how similar each subsequence is to its nearest neighbor in another region of the song. Adopting a small threshold to the distance between subsequences, we can use SiMPle to define these jumps. These characteristics may be explored in order to create a player for endless reproduction. We refer the interested reader to the supporting website ([SILVA *et al.*,](#)) for examples of this functionality.

5.5.5 Sampling Identification

In addition to providing a global distance estimation between different songs, SiMPle is also powerful in examining local similarities. An interesting application that may exploit this ability is the automatic identification of samples. Sampling is the act of “borrowing” the instrumental basis or main melody from another song. This is a common approach in electronic and hip-hop music.

In contrast to cover versions, sampling is used as a virtual “instrument” to compose new songs. However, algorithms that look only for local patterns to identify versions of the same track may classify a recording using samples as a cover song. Using SiMPle, we can discover that the sampling excerpts have small distance values. In contrast, the segments related to the new song have significantly higher values.

Figure 58 shows an example of the usage of SiMPle to spot sampling. In this case, we compare the song “Under Pressure” by Queen and David Bowie with “Ice Ice Baby” by Vanilla Ice. Most of the continuous regions with values lower than the mean refer to the sampling of the famous bass line of the former song.

Figure 58 – SiMPle (in blue) obtained between the songs “Ice Ice Baby” and “Under Pressure.” The continuous regions below the mean value (in red) represent the excerpts sampled by Vanilla Ice from Queen’s song

Source: Elaborated by the author.

5.6 Conclusion

In this paper, we introduced a technique to exploit subsequences joins to assess similarity in music. The presented method is very fast and requires only one parameter that is intuitively set in music applications.

While we focused our evaluation on the cover song recognition, we have shown that our approach has the potential for applications in different MIR tasks. We intend to further investigate the use of matrix profiles in the tasks discussed in Section 5.5 and the effects of different features in the process.

The main limitation of the proposed method is that the use of only one nearest neighbor may be sensitive to hubs, i.e., subsequences that are considered the nearest neighbor of many other snippets. In addition, SiMPle cannot be directly used to identify regions where several subsequences are next to each other, composing a dense region. For this reason, we intend to measure the impact of the reduction in the amount of information in different tasks. Given these limitations, we plan to explore how to incorporate additional information to SiMPle with no loss of time and space efficiency.

We notice that we are committed to the reproducibility of our results, and we encourage researchers and practitioners to extend our ideas and evaluate the use of the SiMPle in different MIR tasks. To this end, we have created a website ([SILVA et al.,](#)) with the complete source code used in our experiments and videos highlighting some of the results presented in this work.

ELASTIC TIME SERIES MOTIFS AND DISCORDS

Abstract: Time efficiency has been almost exclusively the focus of research papers on finding patterns in streaming time series data. Meanwhile, these papers usually neglect common issues in such kind of data, like distortions on the time axis. For this reason, the quality of the patterns found by state-of-the-art algorithms may be affected in trade of fast computation. In this paper, we present a method for calculating a representation of the subsequences similarities under the Prefix and Suffix Invariant Dynamic Time Warping distance that is able to reveal a variety of patterns in the data, being that we focus our evaluation on motifs and discord discovery. Providing invariance to warping and spurious endpoints caused by the segmentation of subsequences, we encounter interesting subsequences which may be distorted and have different lengths, increasing the quality of the discovered patterns. In addition, we propose a suite of simple methods to speed-up our algorithm, making it more than one order of magnitude faster than a straightforward implementation of motifs and discords discovery under Dynamic Time Warping. Finally, our method can be interrupted at any time, providing a good approximate solution faster than the state-of-the-art algorithm for motif discovery under Euclidean distance.

6.1 Introduction

In the face of a considerable growth of applications based on temporal data, finding *typical* and *atypical* patterns in streaming time series has been a common focus of research. Specifically, approximately repeated shapes (*motifs*) and subsequences that differ from the other patterns in the data (*discords*) have applications in several time series mining tasks, such as classification, clustering, association rule discovery and anomaly detection (KEOGH *et al.*, 2007; MUEEN; KEOGH, 2010).

Virtually all research efforts on finding such patterns utilize the Euclidean distance (ED), mainly motivated by the creation of fast algorithms (KEOGH *et al.*, 2007; MUEEN, 2014; YEH *et al.*, 2016). However, the ED is based on some assumptions that may hinder the discovery of interesting patterns. For instance, it requires the patterns to have the same duration and be perfectly aligned in time. On the other hand, several time series suffer from distortions in the time axis for which ED is not able to deal with. One such example are the time series obtained by tracking human actions. When a human being performs the same movement several times, it is very unlikely that he or she executes perfectly alignment moves. Instead, the movements tend to suffer from *warping* caused by different paces of the same action.

In these cases, it is necessary to use a more flexible distance measure, which allows the non-linear alignment between the observations of the time series. For this purpose, a common practice is to adopt the Dynamic Time Warping (DTW) distance. Despite its proven suitability for tasks like similarity search and nearest neighbor classification in a wide number of application domains (WANG *et al.*, 2013), the DTW is little exploited in motifs and discord discovery.

Presumably, the main reason for this fact is the high computational complexity of DTW. While several methods to speed up DTW in different tasks have been proposed (RAKTHAN-MANON *et al.*, 2012; SILVA; BATISTA, 2016b), accelerate the motif discovery under DTW is still taking its first steps, such as with the proposal of specific indexing structures (TRUONG; ANH, 2015).

Apart from that, there is an important issue that neither ED or DTW are designed to cope. When searching for motifs or discords, the user needs to set the length of the desired candidate subsequences. In addition to be too sensitive to such a parameter (TANG; LIAO, 2008), algorithms which assess subsequences using a fixed-length window – which is the usual procedure (MUEEN, 2014) – leads the distance calculation to suffer from the presence of *prefix* and *suffix* (or spurious endpoints) (SILVA; BATISTA; KEOGH, 2016), i.e., values that belong to the previous and the following patterns or classes.

The presence of such spurious endpoints directly affects the motifs and discords discovery. When two subsequences are similar, but one of them contains a prefix or suffix from another pattern, they are very likely to be considered distant, i.e., they may not be considered motifs. In contrast, a subsequence that is similar to others except by some values that should not be part of the evaluated pattern may be considered a time series discord. In both cases, ignoring values in the extremities allows a more accurate pattern discovery.

In this paper, we propose the use the Prefix and Suffix Invariant DTW (ψ -DTW) (SILVA; BATISTA; KEOGH, 2016) in order to find “elastic” motifs and discords, i.e., with warped patterns and possibly with different lengths. In addition, we propose a set of techniques to speed up the algorithm to find them.

Figure 59 illustrates the motivation for the use of ψ -DTW. Both subsequences represent the same event but are naturally warped, hindering a good distance estimation by the ED. In addition, despite the differences in the endpoints look insignificant, the DTW distance is almost the double than the ψ -DTW. It means that the cost of matching between the observations in the extremities – avoided by ψ -DTW – represent nearly half of the distance obtained by DTW.

Figure 59 – Despite representing the same gesture, these two subsequences are clearly *warped* and have differences in the endpoints, which may interfere in the distance calculation

Source: Elaborated by the author.

In summary, the advantages of our method are:

- it is designed to find warped patterns and is invariant to suffix and prefix, avoiding false positives caused by such issues;
- it finds patterns with different lengths in only one pass through the time series;
- it is more than one order of magnitude faster than a straightforward DTW implementation;
- it can be easily extended for different time series premises and other definitions of motifs and discords;
- it is anytime and gives a good approximate solution at the required runtime. We show that our method is able to find most of the best pair motifs faster than the state-of-the-art ED-based motifs discovery algorithm.

6.2 Definitions and Background

Before introducing our method, we will introduce some definitions and briefly present the necessary background. We begin by the definition of time series.

Definition A *time series* T is an ordered set of values $t = (t_1, t_2, \dots, t_N)$ such that $t_i \in \mathbb{R}$ are referred as *observations*. The number of observations N is referred as the *length* of the time series.

Given the definition of time series, we are in the position to define a subsequence.

Definition A subsequence $t_{q,m}$ is a continuous subset of T of length m starting from the observation q , i.e., $t_{q,m} = (t_q, t_{q+1}, \dots, t_{q+m-1})$, such that $q \in [1, N - m + 1]$.

In this paper, we focus on two types of patterns used in time series mining: discords and motifs. The definition of both primitives directly depends on a distance measure, defined as following.

Definition A distance measure is a function $dist(t_{q,m}, t_{p,m})$ between two subsequences $t_{q,m}$ and $t_{p,m}$ that returns a non-negative real value, which is said the distance between $t_{q,m}$ and $t_{p,m}$.

It is important to notice that the definitions of motifs and discord will not require the chosen distance to be a *metric*, i.e., the distance will need to obey all the properties to be a considered a metric – namely, non-negativity, identity of indiscernibles, symmetry and triangle inequality. The distance measure used by our method respects all the properties except the triangle inequality.

Given the definition of distance measure, we are able to define a nearest neighbor.

Definition The *nearest neighbor* (NN) of a subsequence $t_{q,m}$ is the subsequence $t_{p,m}$ such that $t_{p,m}$ is the (non-trivial) subsequence with the smaller distance measure to $t_{q,m}$. Formally, $NN(t_{q,m}) = t_{p,m} \Leftrightarrow dist(t_{q,m}, t_{p,m}) = \min(dist(t_{q,m}, t_{k,m}))$, such that $k \in [1, N - m + 1]$ excluding trivial matches.

As stated, it is necessary to invalidate trivial matches as candidates of nearest neighbors. A trivial match is defined next.

Definition Given two subsequences $t_{q,m}$ and $t_{p,m}$, they constitute a *trivial match* if $|p - q| \leq \xi$, $\xi \in \mathbb{N}$.

Figure 60 illustrates the concepts defined so far.

Figure 60 – Illustration of a time series with a given subsequence (blue), its nearest neighbor (red) and two examples of trivial matches (green, translated for better visualization)

Source: Elaborated by the author.

We continue with the definition of time series discord (KEOGH *et al.*, 2007).

Definition A *discord* $t_{d,m}$ of the time series T is the subsequence with the largest distance to its nearest neighbor. Formally,

$$\text{dist}(t_{d,m}, \text{NN}(t_{d,m})) = \max(\text{dist}(t_{q,m}, \text{NN}(t_{q,m}))), q \in \{1 \dots N - m - 1\}.$$

There is (at least) two definitions of time series motifs in the literature (MUEEN, 2014). In this work, we use the “similarity-based” or “best-pair” motifs.

Definition The *best pair motifs* are a pair of subsequences $(t_{q,m}, t_{p,m})$ from the time series T which returns the smallest distance among every pair of subsequences with no trivial match. Formally, $\text{dist}(t_{q,m}, t_{p,m}) = \min(\text{dist}(t_{k,m}, t_{l,m})), l, k \in 1 \dots N - m - 1$ and $|l - k| > \xi$.

In this case, the motif discovery is intended to find patterns that are so similar that are very unlikely to occur at random (MUEEN; KEOGH, 2010). For the case in which the user requires more than one faithfully repeated pattern, we define the K -th motif pair.

Definition The K -th *motif pair* is the pair of subsequences $(t_{q,m}, t_{p,m})$ with the smallest distance in the dataset with no overlap with subsequences from any I -th motif pair such that $I < K$.

Despite the other definition of motifs (known as k -motifs or support-based motifs) being commonly used in the literature, we keep only one definition for the sake of simplicity. However, our method is easily extended to find k -motifs. The only necessary change is the substitution of the *best-so-far* distance (c.f. Section 6.3) by the maximum allowed distance – an input parameter for the k -motifs (CHIU; KEOGH; LONARDI, 2003) – as threshold for pruning.

Most of the presented definitions are directly or indirectly related to a distance measure between distinct subsequences. The most common distance measure to search for motifs and discord is the (squared) Euclidean distance (ED), defined by the Equation 6.1.

$$ED(t_{q,m}, t_{p,m}) = \sum_{k=0}^{m-1} (t_{q+k} - t_{p+k})^2 \quad (6.1)$$

The ED measures the dissimilarity by a linear alignment between the subsequences, i.e., by matching observations in the same position in the time axis. For this reason, the ED is very sensitive to small distortions in the time axis (KEOGH; RATANAMAHATANA, 2005), commonly referred as *warping*. It may be caused by different paces, such as two subjects walking on different speeds, the tempo of a music piece or utterances, longer or shorter gestures, among others. Despite some warped patterns should be considered similar, the ED is not able to match them in a proper alignment of the observations.

Many applications require a more flexible matching of observations, in which an observation of the subsequence $t_{q,m}$ at time k can be associated with an observation of the subsequence

$t_{p,m}$ at time $l \cong k$. Among different distance measures proposed in the literature which allows such kind of *elastic* alignment, the Dynamic Time Warping (DTW) is arguably the most relevant and widely used (WANG *et al.*, 2013).

DTW is usually calculated using a dynamic programming algorithm, its recurrence relation is defined by Equation 6.2. For initialization purposes, it is necessary to set $dtw(0,0) = 0$ and $dtw(i,j) = \infty$ for every other cases such that $i = 0$ or $j = 0$.

$$dtw(i,j) = c(t_{q+i-1}, t_{p+j-1}) + \min \begin{cases} dtw(i-1, j) \\ dtw(i, j-1) \\ dtw(i-1, j-1) \end{cases} \quad (6.2)$$

where $i, j = 1 \dots m$ and $c(t_{q+i-1}, t_{p+j-1})$ is the cost of matching two observations t_{q+i-1} and t_{p+j-1} , i.e., the i -th and j -th observation of the subsequences $t_{q,m}$ and $t_{p,m}$, respectively. The matching cost is usually calculated with squared difference between the observations.

A commonly applied technique within the DTW algorithm is the *warping window* or *warping band* (SAKOE; CHIBA, 1978). In a practical standpoint, it limits the distance in the time axis for matching a pair of observations. Specifically, it adds the constraint $|i - j| \leq w$ to the recurrence relation (Equation 6.2), in which the parameter w is referred as warping size or warping length. In addition to speed up the DTW calculation, because it avoids several steps of the recurrence relation, the use of warping windows usually provides better efficacy in tasks like clustering and classification (RATANAMAHATANA; KEOGH, 2005).

Finally, the DTW distance is defined by the cost of aligning the last pair of observations, i.e., $DTW(t_{q,m}, t_{p,m}) = dtw(m, m)$. Figure 61 illustrates how the ED and the DTW align the observations in order to calculate a distance between two subsequences. With the linear alignment obtained by the ED, the slightly displaced valley and the substantially different suffixes make these data be considered distant in the space of subsequences. On the other hand, the non-linear alignment obtained by DTW is robust to the warped valley, but still suffers from the differences in the endpoints.

Figure 61 – Alignment obtained by matching two subsequences by the ED (*left*) and the DTW (*right*)

Source: Elaborated by the author.

The DTW is based on three constraints: endpoints, monotonicity and continuity (KEOGH; RATANAMAHATANA, 2005). The endpoints constraint requires the matching of all obser-

variations, starting at the pair (t_q, t_p) and finishing at (t_{q+m-1}, t_{p+m-1}) . Recently, researchers have noticed the negative impact of such restriction to the time series classification (SILVA; BATISTA; KEOGH, 2016). When segmenting the time series in subsequences, it is usual to observe values from different classes in the same subsequence. In general, few “undesirable” values occur at the first and/or last observations of the segmented time series, which may have severe consequences to the distance calculation.

The Prefix and Suffix Invariant DTW (ψ -DTW) was proposed to circumvent such a problem. The distance measure is based on a subtle modification of the DTW endpoints constraints, allowing the algorithm to skip the matching of a limited number of values among the first and last observations from the subsequences under comparison. Specifically, it requires a parameter to set the maximum number of observations that can be skipped, referred as *relaxing factor*.

While the recurrence relation of ψ -DTW is the same than the traditional DTW (Equation 6.2), its initialization needs to be replaced by the initial condition presented by Equation 6.3.

$$dtw(i, j) = \begin{cases} \infty, & \text{if } (i = 0 \text{ and } j > r) \text{ or } (j = 0 \text{ and } i > r) \\ 0, & \text{if } (i = 0 \text{ and } j \leq r) \text{ or } (j = 0 \text{ and } i \leq r) \end{cases} \quad (6.3)$$

where $r \in \mathbb{N}$ is the relaxing factor. Finally, Equations 6.4 and 6.5 define the ψ -DTW between two subsequences, given a relaxing factor r .

$$\psi\text{-DTW}(t_{q,m}, t_{p,m}) = \min_{i,j \in finalSet} dtw(i, j) \quad (6.4)$$

$$finalSet = \{(m - c, m)\} \cup \{(m, m - c)\} \forall c \in [0, r] \quad (6.5)$$

In addition to reducing the impact of prefixes and suffixes on the distance calculation, ψ -DTW allows the matching of patterns with different lengths when assessing fixed-length subsequences. Specifically, the largest difference in terms of length occurs when ψ -DTW matches all the m observations from one subsequence to $m - 2r$ observations from the other.

Figure 62 illustrates the matching of observations obtained by ψ -DTW. Note that it disregards several of the first observations of the blue (bottom) subsequence, while it considers few observations on the red (top) subsequence as a suffix.

We notice that some research efforts were done on finding motifs with different lengths (TANG; LIAO, 2008; LIN; LI, 2010; NUNTHANID; NIENNATTRAKUL; RATANAMAHATANA, 2012). However, the available methods are slow or only able to find approximate solutions. Our method finds the optimal matching of different lengths in a range defined by the relaxing factor in one single pass. In addition, we propose a suite of methods to make it fast. These methods are presented next.

Figure 62 – The ψ -DTW allows an elastic matching in which a non-linear alignment is obtained and points in the extremities can be ignored if they do not present a similar pattern

Source: Elaborated by the author.

6.3 Proposed Method

The main motivation for our method is the recent proposal of the matrix profile (MP) (YEH *et al.*, 2016). The MP is a novel representation of the subsequences similarities in a time series which has direct implications on motifs, discord and shapelets discovery, as well as in semantic segmentation, density estimation, and other tasks.

The MP is essentially a vector with $N - m + 1$ values which stores the distance from each subsequence to its nearest neighbor. With such simple structure, a quick linear search finds several different patterns in the time series, including motifs and discords. Specifically, the smallest value in the MP (which is repeated at least once) points to a motif and the highest value points to the discord.

The MP is accompanied by a vector containing the pointers to the nearest neighbor of each subsequence in the time series, referred as MP index. For clarity, consider that $t_{p,m}$ is the NN of the subsequence $t_{q,m}$. While the MP in the position q stores the value obtained by $ED(t_{q,m}, t_{p,m})$, the MP index stores p .

In this work, we develop a method to find motifs and discords that may suffer from warping and may have slightly different lengths, which we call *elastic motifs* (ELMO) and *elastic discords* (ELD). For this purpose, we construct a matrix profile on the ψ -DTW, which we refer as *elastic distance matrix profile* (EMP).

In a practical standpoint, the construction of the EMP is a succession of similarity searches. When looking for nearest neighbors of a query under DTW, we can make use of the fastest tool for the exact similarity search under DTW known so far (RAKTHANMANON *et al.*, 2012). When searching for ELMO and ELD, we need to adapt the techniques employed by such tool in order to deal with ψ -DTW. In addition, we propose two techniques that are specific for the matrix profile constructions – instead of solely the similarity search – and briefly discuss how to generalize our method for different definitions of motifs and discord.

6.3.1 Online Normalization

When searching for patterns on time series, several invariances are required (BATISTA *et al.*, 2014). As earlier discussed, ψ -DTW is a distance measure to provide *invariance* to spurious endpoints. Two other invariances are especially relevant on comparing subsequences: amplitude and offset, which may be suppressed by the normalization or standardization of the data. For that reason, we are assuming in this work that the subsequences need to be normalized. Specifically, we use the z-score to provide such invariances. The *z-normalization* procedure transforms the subsequence such that its mean is 0 and its standard deviation is 1.

A straightforward implementation of the z-normalization could drastically increase the efficiency of the search. Instead of calculating the mean and standard deviation for each subsequence, we may keep the sum and the squared sum of the observations to calculate the necessary statistics for the normalization process. For this purpose, we need to define the mean and the variance – the squared standard deviation – in terms of such summations. Equation 6.6 defines such statistics.

$$\mu = \frac{1}{m} \left(\sum_{i=q}^{m-1} t_i \right) \quad \sigma^2 = \frac{1}{m} \left(\sum_{i=q}^{m-1} t_i^2 \right) - \mu^2 \quad (6.6)$$

Once we have the mean and variance for the subsequence $t_{q,m}$, such values do not need to be recalculated for $t_{q+1,m}$ from the scratch. Instead, we subtract the observation t_q and add t_m to the summations. Finally, the mean and the standard deviation may be calculated in constant time by replacing the summation factors – in parenthesis – of Equation 6.6 by the stored values.

6.3.2 Lower Bounding

The most relevant technique to speed up the similarity search under DTW is arguably the lower bounding. Before introducing the lower bounding technique, we need to specify the *best-so-far* distance (*bsf*). Once we are interested in the NN of each subsequence, we can store the DTW distance to its nearest neighbor known up to a certain moment of the algorithm execution in the variable *bsf*. The main purpose of storing such value is to avoid the expensive DTW algorithm, discarding its calculation to subsequences which we may know *a priori* to not be the NN. In other words, with the *bsf*, we are able to restrict the space of candidates for NN.

For this purpose, we make use of lower bound (LB) functions. An LB is given by a function – with low computational complexity – which returns a real value that is guaranteed to be lower or equal the DTW between two objects. If the LB between two subsequences provides a value that is greater than the *bsf*, so clearly the DTW between such a pair of subsequences is also greater than the *bsf*. In this case, the DTW algorithm does not need to be used.

The ψ -DTW has an LB function, based on the widely used LB_Keogh for DTW (KEOGH; RATANAMAHATANA, 2005), which we refer in this work as LB_ψ . The calculation of LB_ψ

consists of two main steps. Given a subsequence $t_{q,m}$, the first step is the calculation of two *meta*-subsequences of length m which “wraps” $t_{q,m}$ according the allowed warping. Such meta-subsequences compose an upper and lower envelopes around the subsequence under analysis, $U = u_1, u_2, \dots, u_m$ and $L = l_1, l_2, \dots, l_m$, respectively. The envelopes are defined by Equation 6.7.

$$U_i = \max_{i-w \leq j \leq i+w} t_{q+j} \quad L_i = \min_{i-w \leq j \leq i+w} t_{q+j} \quad (6.7)$$

It is important to notice that both envelopes may be calculated only once at the beginning of the EMP calculation. After that, for each subsequence, we only need to “adjust” the meta-subsequences according the z-normalization (c.f. Section 6.3.1). With such envelopes, the LB is given by Equation 6.8.

$$LB_{\psi}(t_{q,m}, t_{p,m}) = \sum_{i=r+1}^{m-r} \begin{cases} (t_{p+i} - U_i)^2, & \text{if } t_{p+i} > U_i \\ (t_{p+i} - L_i)^2, & \text{if } t_{p+i} < L_i \\ 0, & \text{otherwise} \end{cases} \quad (6.8)$$

Figure 63 illustrates the calculation of LB_{ψ} .

Figure 63 – The LB_{ψ} is given by the Euclidean distance between the assessed subsequence and the envelope around the query in the region that cannot be disregarded according the relaxing factor

Source: Elaborated by the author.

The LB_{ψ} is not a symmetric function, i.e., $LB_{\psi}(t_{q,m}, t_{p,m}) \neq LB_{\psi}(t_{p,m}, t_{q,m})$. Therefore, if $LB_{\psi}(t_{q,m}, t_{p,m})$ is not enough to prune the distance calculation, we repeat the LB calculation reversing the subsequences. In other words, we consider the envelopes around $t_{p,m}$ and calculate $LB_{\psi}(t_{p,m}, t_{q,m})$. The ψ -DTW calculation is required only in the case when both LB are lower than the *bsf*.

6.3.3 Early Abandoning

The LB is calculated by an iterative algorithm which accumulates the distance between the envelopes and the currently assessed subsequence (c.f. Equation 6.8). For this reason, the partial values of such algorithm are monotonically increasing. Once it achieves any value that is greater than the *bsf*, it is guaranteed that the distance between the subsequences under comparison can be discarded. When that happens, we can admissibly abandon the LB calculation.

The early abandoning during the LB calculation is the responsible for most of the pruning during the search for the NN. When the *bsf* is achieved in few steps, in addition to pruning the DTW algorithm, the decision to abandon is even faster.

For this reason, we sort the LB calculation according to the absolute value of the subsequence. It is because the values with high amplitude are likely to contribute with a higher value in the LB. Then, the decision to prune is usually taken in fewer steps than if it was done in the “natural” order of the observations.

When the distance calculation is required, a similar strategy can be used during such a procedure. Given that the partial alignments obtained by DTW (as well as ψ -DTW) have a monotonically increasing cost, once $dtw(i, j) \forall i - w \leq j \leq i + w$ is greater than the *bsf*, the distance calculation can be abandoned. In addition to that, we can use the contributions of the lower bound to improve the DTW early abandoning power. Specifically, we use the value $\max_{i-w \leq j \leq i+w} (dtw(i, j)) + lb[t_{q+i+1, m-i-1}, t_{p+i+1, m-i-1}]$ as decision threshold for abandoning, where *lb* is a vector containing the cumulative costs of the LB.

6.3.4 Exploring the Symmetry of ψ -DTW

The effectiveness of lower bounding and early abandoning depends on one major factor: the value of the best-so-far distance. Specifically, if a good (small) *bsf* is found in early stages of the search, such techniques will prune most of the candidates for NN. On the other hand, if the distance to the subsequence is somehow ordered in the search, the pruning power is too low. In other words, the *bsf* works as a threshold for pruning. Making such value quickly become small induces a faster procedure for ELMO and ELD discovery.

The fact that ψ -DTW is a symmetric distance measure may be useful for the EMP calculation. Consider we are currently searching for the NN of $t_{q,m}$. For every subsequence $t_{p,m}$ for which the LB and the early abandoning were not able to prune, we can verify the value stored in the *i*-th position of the EMP. If such value is greater than ψ -DTW($t_{q,m}, t_{p,m}$), we update the EMP and EMP-index in such position.

Using this simple technique, we provide a reasonable guess of a *bsf* to the subsequence $t_{p,m}$. When searching for its NN, having such a value since the beginning improves the pruning power of the LB and early abandoning techniques. In addition, we make the calculation of ψ -DTW($t_{q,m}, t_{p,m}$) unnecessary.

6.3.5 Heuristic Order

As stated before, a good initial guess of the *bsf* tends to improve the runtime of the similarity search. Following this reasoning, we also propose a heuristic choice of the subsequence to be assessed in each step.

While a straightforward implementation of ELMO and ELD discovery would follow the natural order of the subsequences, we use the EMP information to choose the next subsequence as the one with best initial *bsf*. Specifically, while searching for the NN of the currently assessed subsequence, we store the position of the subsequence with the smallest value in the EMP which was not assessed yet. Then, in the next iteration, our method will skip to such a subsequence.

The advantages of this method are twofold: it reduces the runtime for the calculation of the EMP and it improves the efficacy of the anytime motifs discovery (c.f. Section 6.5.3).

6.4 Why ψ -DTW Instead of the Regular DTW?

A question that may commonly arise at this point is “*why should one choose a DTW variation instead of the traditional algorithm?*” The answer to this question is based on the fact that the traditional DTW is actually a special case of the ψ -DTW. Specifically, when comparing two subsequences with relatively similar endpoints, the distance obtained by both methods is the same. On the other hand, when prefixes and suffixes have a higher impact on the distance calculations, ψ -DTW is able “fix” the DTW by ignoring these spurious endpoints.

As a practical consequence, the traditional distance function may discard several relevant motifs by considering them distant, given that their endpoints are disproportionately contributing to the total distance. As an example, consider the pair of subsequences presented in Figure 64. This pair is one of the most relevant motif pairs found by ψ -DTW in the athlete positioning dataset (c.f. 6.5.2.1). The same subsequences constitute a pair of nearest neighbors according to the traditional DTW. However, the DTW considers them a distant pair in relation to other matches. In fact, while the ψ -DTW between the subsequences is among the smallest 5% of the values in the EMP, the DTW distance between them is not even in the first quartile of the DTW-MP.

Figure 64 – A pair of motifs found by ψ -DTW considered distant by the traditional DTW

Source: Elaborated by the author.

On the opposite side, the DTW may consider two subsequences dissimilar when only the endpoints have a high discrepancy. In general, this fact represents cases in which the endpoints are not part of the pattern. This makes the DTW “miss” the correct time series discords. Figure 65 shows the third discord found by DTW (with its nearest neighbor) and its nearest neighbor according to ψ -DTW in the athlete positioning dataset. While the DTW distance between the subsequences is among the 0.1% of the greatest distances in the MP, around 10% of the values in the EMP are greater than the ψ -DTW between the presented subsequences. Note that, except by

a clear prefix and some discrepant observations at the end, both subsequences describe a similar behavior. As the result, that subsequence is only considered the tenth discord by ψ -DTW.

Figure 65 – A discord found by DTW (red) and its nearest neighbor (blue) according the traditional DTW (top) and ψ -DTW (bottom)

Source: Elaborated by the author.

6.5 Experimental Evaluation

In this section, we evaluate the scalability and the quality of the solutions obtained by our method. For the sake of reproducibility, we developed a website for this work¹, where we make available source codes, datasets, and detailed results.

It is important to notice that evaluating the significance of time series motifs and discords is a difficult matter. Once the discover of these patterns is an unsupervised task, it is impossible to use evaluation measures which depend on a ground truth, such as accuracy and F-measure. Despite there is a few work on how to evaluate the significance of the discovered motifs (CAS-TRO; AZEVEDO, 2011), the usual practice is the visual inspection of the subsequences. In our experimental evaluation, we try as much as possible to use subjective ways to evaluate the presented motifs and discords.

6.5.1 Scalability

In order to evaluate the scalability of our method, we measured the runtime² of the EMP calculation for random walk data varying two parameters: the length of the time series (N) and the length of the subsequence (m). We fixed the values of the warping window length (w) and the relaxing factor (r) as 5% of N . This is based on a strong evidence that small

¹ <https://sites.google.com/view/elastic-motifs-discords>

² All the experiments were taken on the same computer – a 40-core Intel®Xeon®CPU E5-2690 v2 @ 3.00GHz with 130 MB of RAM memory– with no other but operating system-related processes running in parallel.

windows (usually smaller than 10%) are a good choice for most cases of nearest neighbor search (RATANAMAHATANA; KEOGH, 2005).

For the sake of comparison, we also annotated the runtime of a brute-force method for the DTW-based MP calculation, i.e., without the optimizations described in Section 6.3. In addition, we compared the time to calculate the ED-based MP by the Matlab® implementation of STOMP (ZHU *et al.*, 2016) – the state-of-the-art algorithm for MP calculation –, without the parallel computing techniques.

The brute force DTW and the STOMP respectively constitute a topline and a baseline for our method. To calculate the first distance profile, STOMP uses a time proportional to $\mathcal{O}(n \log n)$. For all the remaining subsequences, the algorithm only requires $\mathcal{O}(n)$ operations. Then, the time complexity of STOMP is $\mathcal{O}(n^2)$, what is a very impressive achievement given that it is the minimum number of operations required to assess every pair of subsequences in $\mathcal{O}(1)$. On the other hand, the time complexity for the brute force DTW is $\mathcal{O}(n^2 mw)$, given by the $\mathcal{O}(n^2)$ pairs of subsequences and the $\mathcal{O}(mw)$ complexity of the DTW distance. All the speed up methods presented in Section 6.3 are heuristics and may eventually fail on the indexing of DTW. For this reason, the worst case complexity of our method is the same than the brute force DTW. However, our experimental evaluation shows that, in practice, the runtime of our method is much closer to the baseline than to its worst case, as presented next.

Figure 66 shows the runtimes for a random walk with 100,000 observations and subsequence length varying between 2^5 and 2^9 .

Figure 66 – Runtime for calculating the distance matrix profile using a brute force DTW algorithm (red), our method (blue), and STOMP (green) for different subsequence lengths. The markers point to real runtime values and the lines were obtained by the Spline interpolation (BOOR, 1978)

Source: Elaborated by the author.

As previously stated, the time complexity of STOMP is independent on the subsequence length. It is the reason for the (approximately) constant runtime for STOMP in Figure 66. The runtime of our method increases according to the subsequence length, but we can notice a moderate increasing in comparison to the brute force DTW.

The other experimented parameter, which affects all three methods, is the length of the time series. Figure 67 shows the results for the subsequence length fixed in 128 observations.

Figure 67 – Runtime for calculating the distance matrix profile using a brute force DTW algorithm (red), our method (blue), and STOMP (green), varying the time series length

Source: Elaborated by the author.

Despite the STOMP is faster for long time series and/or subsequences, a direct comparison is not a fair approach. While the ED-based method is faster, it is not suitable for datasets that suffer from warping, mainly for small datasets. When the time series is long, the same pattern presumably occurs several times in the same data stream. In this case, the ED is more likely to find patterns that are “similarly warped,” a scenario that is more suitable for the use of the ED. This makes clear an important trade-off between accuracy and time efficiency.

The main issue of the enunciated trade-off is that there are no clear guidelines for choosing the distance to use. We claim that the choice must be done according to the application. For instance, consider two situations of sports monitoring: soccer player’s positioning (c.f. Section 6.5.2.1) and tennis player’s hand tracking (YABE; TANAKA, 1999). While the position of a soccer player tends to widely vary and suffer from severe warping during a match, the movements of a tennis player’s hand are likely to be faithfully repeated several times during a single game. So, in the former case, we recommend the use of ψ -DTW, while we believe that ED is perhaps enough for the second task.

6.5.2 Case studies

In order to demonstrate the quality of the patterns discovered by our method, we experimented the ELMO and ELD discovery in 5 case studies from different domains. For this purpose, we compared the ED-based motifs and discords to the patterns found by the ψ -DTW with fixed warping length and relaxing factor as 5% of the subsequence length and the trivial matches in a range of one-quarter of the subsequence length.

6.5.2.1 Athlete Positioning

The first dataset used in our experiments tracks a soccer player's trajectory in the attack/defense axis recorded by a ZXY Wearable Tracking sensor³ during an entire match (PETTERSEN *et al.*, 2014). The data was obtained in a sampling rate of 20Hz, resulting in a time series with 114,793 values.

We present the motifs and discords using a subsequence representing 2 minutes of the match, i.e., 2400 observations. We notice, however, that the motifs and discords for this and the other datasets are similar when we resample the time series, which has implications for the time efficiency. For the sake of space, we illustrate this observation on the paper's website.

Figure 68 shows the first ED-motifs and ELMO. In this case, the presence of spurious endpoints in the pair of ELMO is easily noticed. In addition, it is clear the presence of warping, mainly in the interval between observations 500 and 1200. For these reasons, such a pair of subsequences can not be considered as motifs by the ED. In fact, while the ED for the best motif is 11.79, the ED between the pair of ELMO is 18.03.

Figure 68 – First motif pair found by ED (*top*) and the first pair of ELMO (*bottom*) in the athlete positioning data

Source: Elaborated by the author.

Another interesting behavior of the ED in the presence of warped data is that it tends to favor more “conservative” patterns. In this case, it is usual to find motifs like the presented in Figure 68 (*top*), in which flat lines and/or single diagonals compose most part of the pattern. The other side of the coin is that complex subsequences tend to be far from any other patterns in the Euclidean space, which favors such kind of pattern to be considered a discord. This is easily seen by the discovered discord, presented in Figure 69.

The discord found by the ED would be quite similar to its nearest neighbor if the distance measure was invariant to a small warping and the presence of a suffix and prefix. Effectively, if we consider both invariances, it is possible to find even more similar subsequences to the discord. Figure 70 shows the first ED-discord and its nearest neighbor according to ψ -DTW. Note that the subsequences are very similarly shaped, but affected by a subtle warping.

³ <http://chyronhego.com/sports-data/zxy>

Figure 69 – The discord found by ED (red) and its nearest neighbor (blue) in the athlete positioning data

Source: Elaborated by the author.

Figure 70 – The discord found by ED (red) and its nearest neighbor according to the ψ -DTW (blue) in the athlete positioning data. The gray lines show a subset of the alignment between observations obtained by the elastic distance

Source: Elaborated by the author.

On the other hand, when searching for the ELD, we are able to find more reliable exceptional subsequences. Figure 71 verifies such statement. The exhibited ELD corresponds to the concatenation of the last moments of the first half and the beginning of the second half. At that time, the player turns off the sensor and turn it on again when he backs to the pitch, in a different location. For this reason, there is a clear “jump”, impossible to be achieved by a soccer player.

Figure 71 – ELD (red) with its respective nearest neighbor (blue) in the athlete positioning data

Source: Elaborated by the author.

6.5.2.2 Motion Capture

Motion capture technologies have been strongly developed in the last decade for different purposes, allowing precise tracking of human movements regarding temporal and spatial information. In this work, we used the HDM05 MoCap Database (MÜLLER *et al.*, 2007). The session used in our work is the longest of a set of repetitions of walking, sitting and lying down on different surfaces. We chose to track the right wrist because it is arguably one of the

dimensions with more remarkable and wide movements. The subsequence was composed by 300 observations, which represents 2.5 seconds of movements. Figure 72 present the first three pairs of motifs by ED and ψ -DTW.

Figure 72 – Three first motifs according ED (*top*) and first pairs of ELMO (*bottom*) in the motion capture data

Source: Elaborated by the author.

The previously noticed bias for matching conservative patterns when using ED can be easily seen again in these results. Such a bias also have a notable effect on the discord discovery, which can be seen in Figure 73. In this case, the ED tends to consider a simple pattern as NN of the (slightly complex) discord. Instead, the NN according to the ψ -DTW seems to be a very similar pattern to such a subsequence.

Figure 73 – Discord discovered by the ED with its nearest neighbors according ED (*top*) and ψ -DTW (*bottom*) in the motion capture data

Source: Elaborated by the author.

Figure 74 shows the ELD and its nearest neighbor. Note that the ELD is much more simple than the discord presented in Figure 73.

In order to strengthen the evidence of that ψ -DTW is able to find more relevant discords, we ran our method under the only data in the HDM05 MoCap Database with annotated “strong artifacts.” Specifically, such annotation is related to the left leg in a session of activities. So, we used the values collected from the sensor located in the left femur of the subject. In this case, the

Figure 74 – ELD in the motion capture data and its nearest neighbor

Source: Elaborated by the author.

ED does not consider the correct subsequence as discord. On the other hand, the subsequence found as ELD covers part of the annotated anomaly.

6.5.2.3 Gesture Analysis

Gesture analysis has attracted the attention of the scientific community and practitioners in different tasks. In this study, we used the Palm Graffiti Digits dataset (ALON *et al.*, 2009), was obtained by recording different subjects “drawing” digits in the air while facing a 3D camera. The used data consists on the concatenation of every subsequence of hands’ tracking in a random order. In addition, we aggregate the three dimensions of the time series in a single dimensional stream by the sum of their values. The gestures in such a dataset have different duration, but they usually have less than 180 observations. In our experiment, we looked for patterns that could have two gestures, i.e., we used a subsequence of 360 values. Figure 75 presents the results.

Figure 75 – Three first motifs according ED (*top*) and first pairs of ELMO (*bottom*) in the gesture data

Source: Elaborated by the author.

Despite motif discovery is essentially an unsupervised task, we could make use of the labels for a better analysis of our motifs in this case. For both distance measures, the patterns found consists of two gestures of the same class in the same order in every three pairs of motifs. In all the discovered patterns, however, there are suffixes and prefixes from different classes. In this case, the ED tends to match patterns with conservative endpoints.

The discovered discord is presented next, in Figure 76. Similarly to previous results, the discord found by the ED has a very similar pattern if we ignore spurious endpoints and a slight warping.

Figure 76 – Discord according the ED with its nearest neighbor (*top*) and the subsequence considered nearest neighbor according the ψ -DTW (*bottom*) in the gesture data

Source: Elaborated by the author.

Figure 77 presents the discovered ELD. In both ED and ψ -DTW cases, the classes between the discovered discord and its nearest neighbor are divergent.

Figure 77 – ELD in the gesture data and its nearest neighbor

Source: Elaborated by the author.

6.5.2.4 Music Processing

Another application domain in which the data commonly suffers from warping is the music processing. This fact is readily noticed, for instance, in live performances. In this case study, we experimented our method in a live jazz concert data. The time series is composed of features obtained by the Harmonic Change Detection Function (HCDF) ([HARTE; SANDLER; GASSER, 2006](#)). The HCDF measures the tonal centroid variation in the recording. We used 20 HCDF per second, obtaining a total of 81,413 values in a concert that lasts around 1 hour and 8 minutes. The subsequences in this experiment represent 10 seconds, i.e., they are composed of 200 data points. Figure 78 presents the first pairs of motifs.

The visual analysis of HCDF may be difficult, but we can evaluate the motifs according to the audio⁴ and their locations. The first three ED-based motifs have a feature in common: every pair appears in the same song, usually few seconds apart. It means that the ED is not revealing anything but the obvious. A common approach adopted by jazz musicians is to have a

⁴ The website for this paper contains audio files in which is possible to *listen* to the motifs and discords discovered in this dataset

Figure 78 – Three first motifs according ED (*top*) and first pairs of ELMO (*bottom*)

Source: Elaborated by the author.

basis (like repetitions of guitar or bass phrases) and use them to mark solos and improvisations. In the first motif, for instance, the HCDF represents excerpts of the guitar base of the same song, separated by improvisations by different musicians. In the second pair of motifs, the repetition happens in a sequence, i.e., just after the pattern is finished it is played again.

On the other hand, the ELMO pairs represent excerpts from different songs. It means that ψ -DTW is finding more interesting patterns, in which the tonal variations of subsequences from different tracks are similar. Moreover, it is clear the contribution of the warping and spurious endpoint invariances in this case.

Unfortunately, none of the experimented distance measures were able to find relevant/correct discords. It is easily verified by the fact that the discord found by the ED sounds very similar to its NN according ψ -DTW. The same happens to the opposite situation, i.e., between the ELD and its NN according to the ED.

6.5.3 Anytime ELMO discovery

An anytime algorithm is an algorithm that is able to be interrupted and provides a (good) partial solution to the user at any point of its execution. The approximate solution is expected to be the best possible solution given the time the algorithm was executed.

The algorithm proposed in this work is an anytime algorithm. Given that it iteratively calculates the EMP, it may be interrupted at anytime to evaluate the current solution and, if we store a vector indicating which subsequences were already assessed, we can continue the EMP calculation from the same point. In addition, different orders of picking the subsequences allow us to achieve different approximate solutions in the first steps. The order proposed in this work (c.f. Section 6.3.5) privileges the fast ELMO discovery.

To ensure the quality of the anytime property of our method, we performed an experiment using random walk data with 100,000 observations to find patterns using a window of 128 values to compose the subsequences. During such experiment, we wrote down the EMP 10 times equally

spaced in terms of the time series length. In other words, we interrupted the execution after assessing every multiple of 10,000 subsequences.

Despite the first interruption on the algorithm was done after 10% of the time series been assessed, the algorithm arrived such state in around 5% of the total time to achieve the final solution.

In each interruption of our algorithm, we must check the quality of the partial solution. Interestingly, the first 6 pairs of ELMO are the same than the obtained in the final solution. From the next 4 pairs, i.e., from 7 to 10, the partial solution found 2 of them. In summary, from the first 10 best motif pairs, 8 were already found after 5% of the total time.

Figure 79 presents the final and partial solutions. Notice that most of the lowest values were already found by our algorithm. The red values that differ from the blue ones are values obtained by the exploitation of the symmetry, so they are not the final distance value and the respective subsequences will be assessed in a future iteration. In addition, the missing values in the partial solution have no value associated by neither assessing the subsequence or the use of symmetry. They are usually related to high distance values.

Figure 79 – Final EMP (blue) and the partial values obtained after 5% of the total execution time (red)

Source: Elaborated by the author.

It is important to notice that the presented results were obtained after 142 seconds of execution. The method STOMP spent 283 seconds to construct the matrix profile for the Euclidean distance.

Although the results for the ELMO discovery are remarkably good, we could not achieve such outcome for the ELD discovery. Specifically, the correct ELD is found only in the ninth interruption when around 88% of the total runtime has already been spent. For this reason, the anytime characteristic of our method is only suitable for the motif discovery. We intend to study different heuristics in order to improve the anytime discord discovery.

6.6 Conclusion

In this paper, we proposed an approach to construct a subsequences distance matrix profile under the ψ -DTW in order to find motifs and discords on time series data. Our method allows the matching of subsequences with warping and spurious endpoints. We have shown that

our method is able to find more relevant patterns on different domains. In addition, we proposed a suite of methods that is responsible for a relevant speed up of the pattern discovery. Finally, our method is anytime and have shown excellent results on the motifs discovery even faster than the state-of-the-art to calculate the ED-based matrix profile.

As future work, we intend to investigate the proposed methods for different definitions of motifs and discords, as well for different tasks such as shapelet discovery and clustering. In addition, we plan to develop good heuristics in order to improve the anytime feature of our method for different purposes.

OTHER CONTRIBUTIONS

The main topic of this thesis is the time series data mining by similarity-based methods. Although we have presented the most relevant work done in this research, our effort also resulted in other contributions in related applications. An example is the previously discussed Matrix Profile, which generated two publications (YEH *et al.*, 2016; YEH *et al.*, 2017) beyond the music data mining applications described in Chapter 5.

Due to space restrictions, we have included in the text just the use of the Matrix Profile on music analysis. However, we other contributions in this area. In one of these efforts, we have adapted time series shapelets (YE; KEOGH, 2009) to the content-based music information retrieval scenario, with the focus on the cover song recognition (SILVA; SOUZA; BATISTA, 2015). Time series shapelets are small subsequences from the training set which better describe each class for the classification problem. Our proposal adds a training phase to the cover song identification task, to find small excerpts from the feature vectors that best describe each song. We show that we can use such small segments to identify cover songs with higher identification rates and more than one order of magnitude faster than methods used to compare the global similarity, such as the DTW.

Another contribution in the music retrieval area is the work about semi-supervised genre identification by transductive learning (SILVA *et al.*, 2014). In this work, we used a bipartite network representation to perform transductive classification of music, using a bag-of-frames approach to describe the music signals. We showed that this proposal outperforms other music classification methods when few labeled instances are available.

On the time series mining, we investigated how to learn a suitable warping window length for the 1-NN classification under DTW when the number of training examples is limited¹. In these cases, the traditional cross-validation procedures usually fail in finding the optimal value. To circumvent this, we proposed to repeat the cross-validation procedure replacing a fraction

¹ This paper is under review. Please, refer to the list of publications presented at the end of this section.

of the data with synthetically generated time series. In other words, in each repetition of the cross-validation, we used as training set a subset of the original data augmented by a set of synthetic data, obtained by distorting the original objects. The final warping window length is the average of the learned value in all the repetitions. Our results show that this simple procedure can improve the classification accuracy in most of the experimented datasets. In particular, our method tends to provide a higher improvement for the datasets with a few training examples.

We also have collaborated with other researchers in the laboratory. For instance, we have developed a classification approach using ensembles of time series representations. Specifically, we first show that the combination of distances between time series in different representations (e.g., frequency spectrum and Haar wavelets) achieves better results than a single distance in the time domain (GIUSTI; SILVA; BATISTA, 2015). Also, we showed that using these representations to create distance features (similarly to the method proposed by Kate (2016)) provides even better accuracy rates (GIUSTI; SILVA; BATISTA, 2016).

The main purpose to use alternative time series representation is because certain time series features are not evident in the time domain. This fact also motivated the use of recurrence plots – a graphical representation of the recurring patterns in the series – as a data representation for time series classification (SOUZA; SILVA; BATISTA, 2014). We showed that extracting features to describe the texture from this representation and using these features to learn a classification model may overcome the 1-NN under DTW in several different applications.

Finally, we have researched on other data mining domains than time series mining. Firstly, we have contributed to the development of anytime algorithms (LEMES; SILVA; BATISTA, 2014). We proposed a method to improve the SimpleRank (UENO *et al.*, 2006), an algorithm to calculate the relevance of examples for the anytime k -NN classification. Because of SimpleRank results in a significant number of ties, we proposed a method to find a rank examples with the same score by aggregating diversity to these equally ranked objects. In other words, our method makes a “sub-ranking” that spreads the examples with the same score in the space of examples.

Finally, we have contributed to the classification of data streams with concept drift and extreme verification latency (SOUZA *et al.*, 2015b; SOUZA *et al.*, 2015a). In this scenario, we consider that the labels of the arriving new examples are not available. Therefore, we need to model the concept drift in an unsupervised procedure to update the classification model. For this purpose, we used clustering algorithms to track the changes in the space of examples during the stream and associated each partition to a class label.

The following list presents the publications that summarize the contributions described in this thesis.

1. **Silva, D. F.** ; Giusti, R. ; Batista, E. A. P. A. ; Keogh, E. “Speeding Up Similarity Search Under Dynamic Time Warping by Pruning Unpromising Alignments.” Submitted to *Data Mining and Knowledge Discovery*.

2. **Silva, D. F.** ; Yeh, C. M. ; Zhu, Y. ; Batista, E. A. P. A. ; Keogh, E. “Fast Similarity Matrix Profile for Music Analysis and Exploration.” Submitted to the *IEEE Transactions on Multimedia*.
3. **Silva, D. F.** ; Batista, E. A. P. A. “Elastic Time Series Motifs and Discords.” Submitted to the *IEEE International Conference on Data Mining 2017*.
4. Dau, H. A. ; **Silva, D. F.** ; François Petijean; Germain Forestier; Anthony Bagnall; Keogh, E. “Judicious Setting of Dynamic Time Warping’s Window Width Allows More Accurate Classification of Time Series.” Submitted to the *IEEE International Conference on Big Data 2017*.
5. Yeh, C. M. ; Zhu, Y. ; Ulanova, L. ; Begum, N. ; Ding, Y. ; Dau, H. A. ; Zimmerman, Z.; **Silva, D. F.**; Mueen, A. ; Keogh, E. “Time Series Joins, Motifs, Discords and Shapelets: a Unifying View that Exploits the Matrix Profile.” In *Data Mining and Knowledge Discovery*, 2017. p. 1–41. ([YEH et al., 2017](#))
6. **Silva, D. F.**; Batista, G. E. A. P. A.; Keogh, E. “Prefix and Suffix Invariant Dynamic Time Warping.” In: *Proceedings of the IEEE International Conference on Data Mining*, 2016. p. 1209-1214. ([SILVA; BATISTA; KEOGH, 2016](#))
7. **Silva, D. F.**; Yeh, C.-C. M.; Batista, G. E. A. P. A.; Keogh, E. “SiMPle: Assessing Music Similarity Using Subsequences Joins.” In: *Proceedings of the 17th International Society for Music Information Retrieval Conference*, 2016. p. 23-29. ([SILVA et al., 2016](#))
8. **Silva, D. F.**; Batista, G. E. A. P. A. “Speeding Up All-Pairwise Dynamic Time Warping Matrix Calculation.” In: *Proceedings of the SIAM International Conference on Data Mining*, 2016. p. 837-845. ([SILVA; BATISTA, 2016b](#))
9. Yeh, C. M. ; Zhu, Y. ; Ulanova, L. ; Begum, N. ; Ding, Y. ; Dau, H. A. ; **Silva, D. F.**; Mueen, A. ; Keogh, E. “MatrixProfile I: All Pairs Similarity Joins for Time Series: A Unifying View that Includes Motifs, Discords and Shapelets.” In: *Proceedings of the IEEE International Conference on Data Mining*, 2016. p. 1317-1322. ([YEH et al., 2016](#))
10. Giusti, R. ; **Silva, D. F.** Batista, Gustavo E. A. P. A. “Improved Time Series Classification with Representation Diversity and SVM.” In: *Proceedings of the 15th IEEE International Conference on Machine Learning and Applications*, 2016. p. 1-6. ([GIUSTI; SILVA; BATISTA, 2016](#))
11. **Silva, D. F.**; Souza, V. M. A.; Batista, G. E. A. P. A. “Music Shapelets for Fast Cover Song Recognition.” In: *Proceedings of the 16th International Society for Music Information Retrieval Conference*, 2015. p. 441-447. ([SILVA; SOUZA; BATISTA, 2015](#))

12. Souza, V. M. A.; **Silva, D. F.**; Gama, J.; Batista, G. E. A. P. A. “Data Stream Classification Guided by Clustering on Nonstationary Environments and Extreme Verification Latency.” In: *Proceedings of the SIAM International Conference on Data Mining*, 2015. p. 873-881. ([SOUZA et al., 2015b](#))
13. Souza, V. M. A.; **Silva, D. F.**; Batista, G. E. A. P. A.; Gama, J. “Classification of Evolving Data Streams with Infinitely Delayed Labels.” In: *Proceedings of the 14th IEEE International Conference on Machine Learning and Applications*, 2015. p. 214-219. ([SOUZA et al., 2015a](#))
14. Giusti, R. ; **Silva, D. F.** Batista, Gustavo E. A. P. A. “Time Series Classification with Representation Ensembles.” In: *International Symposium on Intelligent Data Analysis*, 2015. p. 108-119. ([GIUSTI; SILVA; BATISTA, 2015](#))
15. **Silva, D. F.**; Rossi, R. G.; Rezende, S. O.; Batista, G. E. A. P. A. “Music Classification by Transductive Learning Using Bipartite Heterogeneous Networks. In: *Proceedings of the 15th International Society for Music Information Retrieval Conference*, 2014. p. 113-118. ([SILVA et al., 2014](#))
16. Souza, V. M. A.; **Silva, D. F.**; Batista, G. E. A. P. A. “Extracting Texture Features for Time Series Classification.” In: *Proceedings of the 22nd International Conference on Pattern Recognition*, 2014. p. 1425-1430. ([SOUZA; SILVA; BATISTA, 2014](#))
17. Lemes, C. I; **Silva, D. F.**; Batista, G. E. A. P. A. “Adding Diversity to Rank Examples in Anytime Nearest Neighbor Classification.” In: *Proceedings of the 13th IEEE International Conference on Machine Learning and Applications*, 2014. p. 129-134. ([LEMES; SILVA; BATISTA, 2014](#))

CONCLUSION

This thesis presented the results obtained with the research on novel and scalable methods for similarity-based analysis of time series data. This work led to contributions in diverse areas since we approached a variety of data mining tasks and different applications domains.

Despite the fact we used the Euclidean distance in some parts of this work (e.g., (YEH *et al.*, 2016; SILVA *et al.*, 2016)), our main concern was the Dynamic Time Warping distance. While it is a very suitable distance measure for a multitude of application domains, its high complexity makes some researchers prefer more efficient distance measures. For this reason, we proposed algorithms to make the time series mining under DTW faster. This research has implications for similarity search, the discovery of motifs and discords, and different tasks that demand a significant amount of DTW calculations.

We note, however, that DTW is not the ultimate algorithm for time series comparison in every application domains. For instance, we have discussed that different applications require different invariances. For this reason, researchers have proposed many variations of DTW. Some examples are the derivative of the raw time series (KEOGH; PAZZANI, 2001) and the aggregation of a positive weight in the matching cost such that distant observations in the time axis receive a higher weight (JEONG; JEONG; OMITAOMU, 2011). In this work, we presented a method to deal with differences in the endpoints of subsequences, a problem which effects seem to be unnoticed so far (SILVA; BATISTA; KEOGH, 2016).

Following this direction, we point as future work a broader and deeper study on the DTW and its effects in different application domains. We intend to investigate how to use different DTW variations together and how it impacts the mining results. In addition to studying new invariances, we aim to combine different strategies to improve the performance of DTW. For instance, we may create a variation of DTW which aggregates the cost weight proposed by Jeong, Jeong and Omitaomu (2011) with invariance to spurious prefix and suffix (SILVA; BATISTA;

KEOGH, 2016) and correction by complexity (BATISTA *et al.*, 2014). The main questions about this approach are the implications on time and accuracy of the combinations.

Another direction for future work is the extension of similarity methods for multidimensional time series. Such an extension has an impact on different domains, such as gesture recognition and trajectory analysis. In these cases, the DTW is widely used and have presented satisfactory results, but several authors highlight its high computational cost (ZHENG, 2015; TOOHEY; DUCKHAM, 2015). Also, the dependence between different dimensions is difficult to model in many cases. Despite the fact that recent work presents adaptive strategies to deal with dimensions dependence in time series classification (SHOKOOHI-YEKTA *et al.*, 2017), there are no reported guidelines for several other mining tasks.

Moreover, novel applications rely on time series in which some of the dimensions are directly related each other while the remaining may be completely independent. Usually, this occurs thanks to the data heterogeneity, such as in the case of time series obtained by sensors attached to professional athletes from different sports (STEIN *et al.*, 2017). In this case, we may record the data from the (two-dimensional) trajectory of the athletes, heart beating, speed, heading, among several other characteristics. However, there is a lack of methods to understand how one dimension may impact the others when we have the interest in discovering knowledge from these data. Besides, different athletes may independently move in the field but, at the same time, they can influence the result of an offensive or defensive action. Finally, this kind of application domain faces a problem with high data dimensionality.

All these observations make the problem of similarity-based trajectory mining a difficult matter. Although we described it on a specific domain, these issues are present on a multitude of applications (ZHENG, 2015). For instance, we could replace “athletes in the field” by “people in touristic landmarks” or “migrating animals.” With this in mind, we will extend this work to the similarity-based trajectory mining.

Finally, we point a direction for future research on music analysis. When developing this research, we noticed a lack on taking advantage from subsequences in music data mining and retrieval. For instance, consider the state-of-the-art procedure for music classification (BERGSTRÄ *et al.*, 2006; SIGTIA; DIXON, 2014; PANTELI; BENETOS; DIXON, 2016). The first step of the procedure is to extract features from short segments of the signal (e.g., 0.01 seconds). After this step, these methods aggregate the features using a larger window (e.g., the equivalent of 5 seconds) by their mean and standard deviation. This procedure transforms each example from a signal to a set of (multidimensional) features. Then, each feature vector composes a new example to learn a classification model. By this approach, if a recording is split in N feature vectors, it will represent N instances in the input for the learning method. When the class of a new recording is required, we transform the signal as described. Each of the obtained feature vectors is classified according to the learned model. Finally, the label of the new recording is given by combining the obtained labels.

Note that such procedure completely ignores the temporal dynamics of the features. Despite this approach aggregates features from consecutive windows, it considers consecutive aggregated features independent each other. Intuitively, varying the features over the time may constitute a valuable information to the classification process. Some authors have pointed to this fact and have used DTW with emotion (DENG; LEUNG, 2015) and rhythm (REN; FAN; MING, 2016) features to achieve better performance on music information retrieval. We intend to investigate how to make use of temporal information on music data mining. Specifically, we aim to use methods to assess the similarity between subsequences (e.g., SiMPle (SILVA *et al.*, 2016)) to improve the performance of music mining and retrieval methods.

BIBLIOGRAPHY

AGGARWAL, C. C.; REDDY, C. K. **Data clustering: algorithms and applications**. Boca Raton, FL, USA: CRC Press, 2013. Citation on page [40](#).

AGHABOZORGI, S.; SHIRKHORSHIDI, A. S.; WAH, T. Y. Time-series clustering – a decade review. **Information Systems**, v. 53, p. 16–38, 2015. Citation on page [41](#).

AGRAWAL, R.; FALOUTSOS, C.; SWAMI, A. N. Efficient similarity search in sequence databases. In: **International Conference on Foundations of Data Organization and Algorithms**. London, UK: Springer-Verlag, 1993. p. 69–84. Citation on page [32](#).

ALON, J.; ATHITSOS, V.; YUAN, Q.; SCLAROFF, S. A unified framework for gesture recognition and spatiotemporal gesture segmentation. **IEEE T Pattern Anal**, IEEE, v. 31, n. 9, p. 1685–1699, 2009. Citations on pages [67](#) e [159](#).

ANGUITA, D.; GHIO, A.; ONETO, L.; PARRA, X.; REYES-ORTIZ, J. L. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In: SPRINGER. **International workshop on ambient assisted living**. Vitoria-Gasteiz, Spain, 2012. p. 216–223. Citation on page [68](#).

ASSENT, I.; WICHTERICH, M.; KRIEGER, R.; KREMER, H.; SEIDL, T. Anticipatory dtw for efficient similarity search in time series databases. **VLDB Endowment**, VLDB Endowment, v. 2, n. 1, p. 826–837, 2009. Citation on page [38](#).

BACHLIN, M.; PLOTNIK, M.; ROGGEN, D.; MAIDAN, I.; HAUSDORFF, J. M.; GILADI, N.; TROSTER, G. Wearable assistant for parkinson’s disease patients with the freezing of gait symptom. **IEEE Transactions on Information Technology in Biomedicine**, IEEE, v. 14, n. 2, p. 436–446, 2010. Citation on page [112](#).

BAGNALL, A.; LINES, J.; BOSTROM, A.; LARGE, J.; KEOGH, E. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. **Data Mining and Knowledge Discovery**, v. 31, n. 3, p. 606–660, May 2017. Citations on pages [38](#) e [39](#).

BARTSCH, M. A.; WAKEFIELD, G. H. Audio thumbnailing of popular music using chroma-based representations. **IEEE Transactions on Multimedia**, IEEE, v. 7, n. 1, p. 96–104, 2005. Citation on page [137](#).

BATISTA, G. E. A. P. A.; KEOGH, E.; TATAW, O. M.; SOUZA, V. M. A. Cid: an efficient complexity-invariant distance for time series. **Data Mining and Knowledge Discovery**, Springer Science & Business Media, v. 28, n. 3, p. 634, 2014. Citations on pages [31](#), [45](#), [60](#), [149](#) e [170](#).

BEGUM, N.; ULANOVA, L.; WANG, J.; KEOGH, E. Accelerating dynamic time warping clustering with a novel admissible pruning strategy. In: ACM. **SIGKDD International Conference on Knowledge Discovery and Data Mining**. Sydney, Australia, 2015. p. 49–58. Citations on pages [41](#), [94](#) e [122](#).

- BELLO, J. P. Measuring structural similarity in music. **IEEE Transactions on Audio, Speech, and Language Processing**, IEEE, v. 19, n. 7, p. 2013–2025, 2011. Citation on page [132](#).
- BEN-DOV, M.; FELDMAN, R. Text mining and information extraction. In: **Data Mining and Knowledge Discovery Handbook**. New York, NY, USA: Springer, 2009. p. 809–835. Citation on page [43](#).
- BERGSTRA, J.; CASAGRANDE, N.; ERHAN, D.; ECK, D.; KÉGL, B. Aggregate features and a da b oost for music classification. **Machine learning**, Springer, v. 65, n. 2, p. 473–484, 2006. Citation on page [170](#).
- BOOR, C. D. **A practical guide to splines**. New York, NY, USA: Springer-Verlag New York, 1978. Citation on page [154](#).
- CAMPOS, R.; DIAS, G.; JORGE, A. M.; JATOWT, A. Survey of temporal information retrieval and related applications. **ACM Computing Surveys (CSUR)**, ACM, v. 47, n. 2, p. 15, 2015. Citation on page [43](#).
- CANDAN, K. S.; ROSSINI, R.; WANG, X.; SAPINO, M. L. sDTW: computing DTW distances using locally relevant constraints based on salient feature alignments. **VLDB Endowment**, VLDB Endowment, v. 5, n. 11, p. 1519–1530, 2012. Citation on page [33](#).
- CANTAREIRA, G. D.; NONATO, L. G.; PAULOVICH, F. V. Moshviz: A detail+overview approach to visualize music elements. **IEEE Transactions on Multimedia**, IEEE, v. 18, n. 11, p. 2238–2246, 2016. Citation on page [138](#).
- CARABIAS-ORTI, J. J.; RODRÍGUEZ-SERRANO, F. J.; VERA-CANDEAS, P.; RUIZ-REYES, N.; CAÑADAS-QUESADA, F. J. An audio to score alignment framework using spectral factorization and dynamic time warping. In: ISMIR. **International Society for Music Information Retrieval Conference**. Málaga, Spain, 2015. p. 742–748. Citation on page [123](#).
- CARRILLO, H.; LIPMAN, D. The multiple sequence alignment problem in biology. **SIAM Journal on Applied Mathematics**, SIAM, v. 48, n. 5, p. 1073–1082, 1988. Citation on page [79](#).
- CASTRO, N.; AZEVEDO, P. J. Time series motifs statistical significance. In: SIAM. **SIAM International Conference on Data Mining**. Mesa, AZ, USA, 2011. p. 687–698. Citation on page [153](#).
- CHAN, K.-P.; FU, A.-C. Efficient time series matching by wavelets. In: **IEEE International Conference on Data Engineering**. Sydney, Australia: IEEE, 1999. p. 126–133. Citation on page [32](#).
- CHÁVEZ, E.; NAVARRO, G.; BAEZA-YATES, R.; MARROQUÍN, J. L. Searching in metric spaces. **ACM Computing Surveys**, ACM, v. 33, n. 3, p. 273–321, 2001. Citations on pages [26](#) e [37](#).
- CHAVOSHI, N.; HAMOONI, H.; MUEEN, A. Debot: Twitter bot detection via warped correlation. In: IEEE. **IEEE International Conference on Data Mining**. Barcelona, Spain, 2016. p. 817–822. Citations on pages [36](#) e [94](#).
- CHEN, N.; LI, W.; XIAO, H. Fusing similarity functions for cover song identification. **Multimedia Tools and Applications**, Springer, p. 1–24, 2017. Citation on page [132](#).

CHEN, Y.; KEOGH, E.; HU, B.; BEGUM, N.; BAGNALL, A.; MUEEN, A.; BATISTA, G. **The UCR Time Series Classification Archive**. <http://www.cs.ucr.edu/~eamonn/time_series_data/>. Accessed 26th Jul, 2017. Citations on pages 52, 54, 63, 87 e 118.

CHIU, B.; KEOGH, E.; LONARDI, S. Probabilistic discovery of time series motifs. In: ACM. **SIGKDD International Conference on Knowledge Discovery and Data Mining**. Washington, DC, USA, 2003. p. 493–498. Citation on page 145.

DAU, H. A.; BEGUM, N.; KEOGH, E. Semi-supervision dramatically improves time series clustering under dynamic time warping. In: ACM. **ACM International on Conference on Information and Knowledge Management**. Indianapolis, IN, USA, 2016. p. 999–1008. Citations on pages 26, 39 e 41.

DEBRAY, A.; WU, R. **Astronomical implications of Machine Learning**. 2013. Citation on page 52.

DEMERDASH, N. A.; BANGURA, J. F. Characterization of induction motors in adjustable-speed drives using a time-stepping coupled finite-element state-space method including experimental validation. **IEEE Transactions on Industry Applications**, IEEE, v. 35, n. 4, p. 790–802, 1999. Citation on page 65.

DENG, J. J.; LEUNG, C. H. Dynamic time warping for music retrieval using time series modeling of musical emotions. **IEEE Transactions on Affective Computing**, IEEE, v. 6, n. 2, p. 137–151, 2015. Citations on pages 43, 116 e 171.

DING, H.; TRAJCEVSKI, G.; SCHEUERMANN, P.; WANG, X.; KEOGH, E. Querying and mining of time series data: experimental comparison of representations and distance measures. **VLDB Endowment**, VLDB Endowment, v. 1, n. 2, p. 1542–1552, 2008. Citations on pages 25, 31, 96 e 122.

FALOUTSOS, C.; RANGANATHAN, M.; MANOLOPOULOS, Y. Fast subsequence matching in time-series databases. In: **SIGMOD International Conference on Management of Data**. Minneapolis, MN, USA: ACM, 1994. p. 419–429. Citations on pages 26, 31, 37 e 47.

FANG, J.-T.; DAY, C.-T.; CHANG, P.-C. Deep feature learning for cover song identification. **Multimedia Tools and Applications**, Springer, p. 1–14, 2016. Citations on pages 43 e 131.

FOOTE, J. Visualizing music and audio using self-similarity. In: ACM. **ACM International Conference on Multimedia**. Orlando, FL, USA, 1999. p. 77–80. Citation on page 124.

FU, Z.; LU, G.; TING, K. M.; ZHANG, D. A survey of audio-based music classification and annotation. **IEEE transactions on Multimedia**, IEEE, v. 13, n. 2, p. 303–319, 2011. Citation on page 131.

GIUSTI, R.; BATISTA, G. E. A. P. A. An empirical comparison of dissimilarity measures for time series classification. In: **Brazilian Conference on Intelligent Systems**. Fortaleza, CE, Brazil: IEEE, 2013. p. 82–88. Citation on page 27.

GIUSTI, R.; SILVA, D. F.; BATISTA, G. E. A. P. A. Time series classification with representation ensembles. **Lecture Notes in Computer Science**, Springer, v. 9385, p. 108–119, 2015. Citations on pages 166 e 168.

_____. Improved time series classification with representation diversity and svm. In: IEEE. **IEEE International Conference on Machine Learning and Applications**. Anaheim, CA, USA, 2016. p. 1–6. Citations on pages [166](#) e [167](#).

GOLDBERGER, A. L.; AMARAL, L. A.; GLASS, L.; HAUSDORFF, J. M.; IVANOV, P. C.; MARK, R. G.; MIETUS, J. E.; MOODY, G. B.; PENG, C.-K.; STANLEY, H. E. Physiobank, physiotoolkit, and physionet components of a new research resource for complex physiologic signals. **Circulation**, Am Heart Assoc, v. 101, n. 23, p. e215–e220, 2000. Citation on page [112](#).

GÓRECKI, T.; ŁUCZAK, M. Multivariate time series classification with parametric derivative dynamic time warping. **Expert Systems with Applications**, Pergamon, v. 42, n. 5, p. 2305–2312, 2015. Citation on page [110](#).

GROSCHÉ, P.; SERRA, J.; MÜLLER, M.; ARCOS, J. L. Structure-based audio fingerprinting for music retrieval. In: ISMIR. **International Society for Music Information Retrieval Conference**. Porto, Portugal, 2012. p. 55–60. Citation on page [132](#).

HALTSONEN, S. An endpoint relaxation method for dynamic time warping algorithms. In: IEEE. **IEEE International Conference on Acoustics, Speech, and Signal Processing**. San Diego, CA, USA, 1984. v. 9, p. 360–363. Citation on page [57](#).

HAN, J.; KAMBER, M.; PEI, J. **Data mining: Concepts and techniques**. Third. San Francisco, CA, USA: Morgan Kaufmann, 2011. Citation on page [40](#).

HARTE, C.; SANDLER, M.; GASSER, M. Detecting harmonic change in musical audio. In: ACM. **ACM workshop on Audio and Music Computing Multimedia**. Santa Barbara, CA, USA, 2006. p. 21–26. Citation on page [160](#).

HJALTASON, G. R.; SAMET, H. Index-driven similarity search in metric spaces (survey article). **ACM Transactions on Database Systems**, ACM, v. 28, n. 4, p. 517–580, 2003. Citation on page [120](#).

HU, B.; CHEN, Y.; KEOGH, E. Time series classification under more realistic assumptions. In: SIAM. **SIAM International Conference on Data Mining**. Astin, TX, USA, 2013. p. 578–586. Citation on page [64](#).

ITAKURA, F. Minimum prediction residual principle applied to speech recognition. **IEEE Transactions on Acoustics, Speech and Signal Processing**, IEEE, v. 23, n. 1, p. 67–72, 1975. Citations on pages [26](#), [28](#), [76](#), [77](#) e [97](#).

JEONG, Y.-S.; JEONG, M. K.; OMITAOMU, O. A. Weighted dynamic time warping for time series classification. **Pattern Recognition**, Elsevier, v. 44, n. 9, p. 2231–2240, 2011. Citations on pages [120](#) e [169](#).

KACHUEE, M.; KIANI, M. M.; MOHAMMADZADE, H.; SHABANY, M. Cuff-less high-accuracy calibration-free blood pressure estimation using pulse transit time. In: IEEE. **IEEE International Symposium on Circuits and Systems**. Lisbon, Portugal, 2015. p. 1006–1009. Citation on page [112](#).

KADOUS, M. W. *et al.* **Temporal classification: Extending the classification paradigm to multivariate time series**. Phd Thesis (PhD Thesis) — University of New South Wales, Sydney, Australia, 2002. Citation on page [68](#).

KATE, R. J. Using dynamic time warping distances as features for improved time series classification. **Data Mining and Knowledge Discovery**, Springer US, v. 30, n. 2, p. 283–312, 2016. Citations on pages 39, 78, 87, 94 e 166.

KEOGH, E. Efficiently finding arbitrarily scaled patterns in massive time series databases. In: SPRINGER. **European Conference on Principles of Data Mining and Knowledge Discovery**. Prague, Czech Republic, 2003. p. 253–265. Citation on page 31.

KEOGH, E.; CHAKRABARTI, K.; PAZZANI, M.; MEHROTRA, S. Locally adaptive dimensionality reduction for indexing large time series databases. **ACM Sigmod Record**, ACM, v. 30, n. 2, p. 151–162, 2001. Citation on page 32.

KEOGH, E.; LIN, J.; LEE, S.-H.; HERLE, H. V. Finding the most unusual time series subsequence: algorithms and applications. **Knowledge and Information Systems**, Springer, v. 11, n. 1, p. 1–27, 2007. Citations on pages 41, 141, 142 e 144.

KEOGH, E.; RATANAMAHATANA, C. A. Exact indexing of dynamic time warping. **Knowledge and Information Systems**, Springer, v. 7, n. 3, p. 358–386, 2005. Citations on pages 26, 27, 47, 53, 71, 99, 145, 146 e 149.

KEOGH, E.; WEI, L.; XI, X.; LEE, S.-H.; VLACHOS, M. LB_Keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures. In: **Very Large Data Bases**. Seoul, Korea: VLDB, 2006. p. 882–893. Citation on page 26.

KEOGH, E.; WEI, L.; XI, X.; VLACHOS, M.; LEE, S.-H.; PROTOPAPAS, P. Supporting exact indexing of arbitrarily rotated shapes and periodic time series under euclidean and warping distance measures. **The VLDB Journal**, Springer-Verlag New York, Inc., v. 18, n. 3, p. 611–630, 2009. Citations on pages 31 e 100.

KEOGH, E. J.; PAZZANI, M. J. Scaling up dynamic time warping for datamining applications. In: **SIGKDD International Conference on Knowledge Discovery and Data Mining**. Boston, MA, USA: ACM, 2000. p. 285–289. Citations on pages 32 e 34.

_____. Derivative dynamic time warping. In: SIAM. **SIAM International Conference on Data Mining**. Chicago, IL, USA, 2001. p. 1–11. Citations on pages 120 e 169.

KIM, S.-W.; PARK, S.; CHU, W. W. An index-based approach for similarity search supporting time warping in large sequence databases. In: IEEE. **International Conference on Data Engineering**. Heidelberg, Germany, 2001. p. 607–614. Citations on pages 47 e 99.

KIYOHARA, T.; ORIHARA, R.; SEI, Y.; TAHARA, Y.; OHSUGA, A. Activity recognition for dogs based on time-series data analysis. In: SPRINGER. **International Conference on Agents and Artificial Intelligence**. Lisbon, Portugal, 2015. p. 163–184. Citation on page 52.

KORN, F.; JAGADISH, H. V.; FALOUTSOS, C. Efficiently supporting ad hoc queries in large datasets of time sequences. v. 26, n. 2, p. 289–300, 1997. Citation on page 32.

KORZENIOWSKI, F.; WIDMER, G. Feature learning for chord recognition: the deep chroma extractor. In: ISMIR. **International Society for Music Information Retrieval**. New York, NY, USA, 2016. p. 37–43. Citation on page 131.

LAMERE, P. **The Infinite Jukebox**. <<http://www.infinitejuke.com>>. Accessed 26th Jul, 2017. Citation on page 139.

- LEMES, C. I.; SILVA, D. F.; BATISTA, G. E. Adding diversity to rank examples in anytime nearest neighbor classification. In: IEEE. **IEEE International Conference on Machine Learning and Application**. Detroit, MI, USA, 2014. p. 129–134. Citations on pages [166](#) e [168](#).
- LIAO, T. W. Clustering of time series data – a survey. **Pattern recognition**, Elsevier, v. 38, n. 11, p. 1857–1874, 2005. Citation on page [40](#).
- LIN, J.; KEOGH, E.; LONARDI, S.; CHIU, B. A symbolic representation of time series, with implications for streaming algorithms. In: ACM. **SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery**. San Diego, CA, USA, 2003. p. 2–11. Citation on page [32](#).
- LIN, J.; KHADE, R.; LI, Y. Rotation-invariant similarity in time series using bag-of-patterns representation. **Journal of Intelligent Information Systems**, Springer, v. 39, n. 2, p. 287–315, 2012. Citation on page [39](#).
- LIN, J.; LI, Y. Finding approximate frequent patterns in streaming medical data. In: IEEE. **IEEE Computer-Based Medical Systems**. Bentley, Australia, 2010. p. 13–18. Citation on page [147](#).
- LINES, J.; BAGNALL, A. Time series classification with ensembles of elastic distance measures. **Data Mining and Knowledge Discovery**, Springer US, v. 29, n. 3, p. 565–592, 2015. Citation on page [38](#).
- LINES, J.; DAVIS, L. M.; HILLS, J.; BAGNALL, A. A shapelet transform for time series classification. In: ACM. **SIGKDD International Conference on Knowledge Discovery and Data Mining**. Beijing, China, 2012. p. 289–297. Citation on page [39](#).
- LIU, N.-H. Effective results ranking for mobile query by singing/humming using a hybrid recommendation mechanism. **IEEE Transactions on Multimedia**, IEEE, v. 16, n. 5, p. 1407–1420, 2014. Citation on page [123](#).
- LONG, X.; YIN, B.; AARTS, R. M. Single-accelerometer-based daily physical activity classification. In: IEEE. **Annual International Conference of the IEEE Engineering in Medicine and Biology Society**. Minneapolis, MN, USA, 2009. p. 6107–6110. Citation on page [38](#).
- LOWE, D. G. Object recognition from local scale-invariant features. In: IEEE. **IEEE International Conference on Computer Vision**. Toronto, Canada, 1999. v. 2, p. 1150–1157. Citation on page [33](#).
- MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. **Introduction to information retrieval**. New York, NY: Cambridge University Press, 2008. Citation on page [43](#).
- MARCACINI, R. M.; CARNEVALI, J. C.; DOMINGOS, J. On combining websensors and dtw distance for knn time series forecasting. In: IEEE. **International Conference on Pattern Recognition**. Cancún, México, 2016. p. 2521–2525. Citation on page [44](#).
- MARCACINI, R. M.; REZENDE, S. O. Incremental construction of topic hierarchies using hierarchical term clustering. In: SEKE. **International Conference on Software Engineering and Knowledge Engineering**. Redwood City, CA, USA, 2010. p. 553. Citation on page [40](#).
- MARTEAU, P.-F. Time warp edit distance with stiffness adjustment for time series matching. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, v. 31, n. 2, p. 306–318, 2009. Citation on page [120](#).

MITSA, T. **Temporal data mining**. Boca Raton, FL, USA: CRC Press, 2010. Citation on page 32.

MOODY, G. B.; MARK, R. G. The impact of the MIT-BIH arrhythmia database. **IEEE Engineering in Medicine and Biology Magazine**, IEEE, v. 20, n. 3, p. 45–50, 2001. Citation on page 112.

MUEEN, A. Time series motif discovery: dimensions and applications. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, Wiley Online Library, v. 4, n. 2, p. 152–159, 2014. Citations on pages 41, 49, 136, 142 e 145.

MUEEN, A.; CHAVOSHI, N.; ABU-EL-RUB, N.; HAMOONI, H.; MINNICH, A. Awarp: Fast warping distance for sparse time series. In: IEEE. **IEEE International Conference on Data Mining**. Barcelona, Spain, 2016. p. 350–359. Citations on pages 35, 36 e 121.

MUEEN, A.; HAMOONI, H.; ESTRADA, T. Time series join on subsequence correlation. In: IEEE. **IEEE International Conference on Data Mining**. Delhi, India, 2014. p. 450–459. Citation on page 37.

MUEEN, A.; KEOGH, E. Online discovery and maintenance of time series motifs. In: ACM. **SIGKDD International Conference on Knowledge Discovery and Data Mining**. Washington, DC, USA, 2010. p. 1089–1098. Citations on pages 141 e 145.

MUEEN, A.; VISWANATHAN, K.; GUPTA, C.; KEOGH, E. **The fastest similarity search algorithm for time series subsequences under Euclidean distance**. 2017. <<http://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html>>. Accessed 26th Jul, 2017. Citations on pages 42, 124, 126 e 127.

MÜLLER, M. Dynamic time warping. In: **Information Retrieval for Music and Motion**. Berlin, Germany: Springer-Verlag Berlin Heidelberg, 2007. chap. 4, p. 69–84. Citations on pages 58 e 129.

MÜLLER, M. **Information Retrieval for Music and Motion**. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007. ISBN 3540740473. Citation on page 96.

MÜLLER, M.; KURTH, F.; CLAUSEN, M. Audio matching via chroma-based statistical features. In: ISMIR. **International Society for Music Information Retrieval Conference**. London, UK, 2005. p. 288–295. Citations on pages 43 e 131.

MÜLLER, M.; RÖDER, T.; CLAUSEN, M.; EBERHARDT, B.; KRÜGER, B.; WEBER, A. **Documentation Mocap Database HDM05**. Bonn, Germany, 2007. Citation on page 157.

MURRAY, D.; LIAO, J.; STANKOVIC, L.; STANKOVIC, V.; HAUXWELL-BALDWIN, R.; WILSON, C.; COLEMAN, M.; KANE, T.; FIRTH, S. A data management platform for personalised real-time energy feedback. In: ECEEE. **International Conference Energy Efficiency in Domestic Appliances and Lighting**. Lucerne, Switzerland, 2015. p. 1293–1307. Citation on page 112.

MYERS, C. S.; RABINER, L. R. A comparative study of several dynamic time-warping algorithms for connected-word recognition. **Bell Labs Technical Journal**, Wiley Online Library, v. 60, n. 7, p. 1389–1409, 1981. Citation on page 57.

NUNTHANID, P.; NIENNATTRAKUL, V.; RATANAMAHATANA, C. A. Parameter-free motif discovery for time series data. In: IEEE. **ECTI International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology**. Phetchaburi, Thailand, 2012. p. 1–4. Citation on page [147](#).

OORD, A. Van den; DIELEMAN, S.; SCHRAUWEN, B. Deep content-based music recommendation. In: NIPS. **Advances in neural information processing systems**. Lake Tahoe, CA, USA, 2013. p. 2643–2651. Citation on page [131](#).

PANTELI, M.; BENETOS, E.; DIXON, S. Learning a feature space for similarity in world music. In: ISMIR. **International Society for Music Information Retrieval Conference**. New York, NY, USA, 2016. Citation on page [170](#).

PARK, C. H. Query by humming based on multiple spectral hashing and scaled open-end dynamic time warping. **Signal Processing**, Elsevier, v. 108, p. 220–225, 2015. Citation on page [116](#).

PETITJEAN, F.; KETTERLIN, A.; GANÇARSKI, P. A global averaging method for dynamic time warping, with applications to clustering. **Pattern Recognition**, Elsevier, v. 44, n. 3, p. 678–693, 2011. Citation on page [78](#).

PETTERSEN, S. A.; JOHANSEN, D.; JOHANSEN, H.; BERG-JOHANSEN, V.; GADDAM, V. R.; MORTENSEN, A.; LANGSETH, R.; GRIWODZ, C.; STENSLAND, H. K.; HALVORSEN, P. Soccer video and player position dataset. In: ACM. **ACM Multimedia Systems Conference**. New York, NY, USA, 2014. p. 18–23. Citations on pages [111](#) e [156](#).

POVINELLI, R. J.; JOHNSON, M. T.; LINDGREN, A. C.; YE, J. Time series classification using gaussian mixture models of reconstructed phase spaces. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 16, n. 6, p. 779–783, 2004. Citation on page [65](#).

RABINER, L.; JUANG, B. Pattern-comparison techniques. **Fundamentals of speech recognition**, p. 141–241, 1993. Citation on page [53](#).

RAFIEI, D.; MENDELZON, A. Similarity-based queries for time series data. In: ACM. **ACM SIGMOD International Conference on Management of Data**. Tucson, AZ, USA, 1997. v. 26, n. 2, p. 13–25. Citation on page [41](#).

RAKTHANMANON, T.; CAMPANA, B.; MUEEN, A.; BATISTA, G.; WESTOVER, B.; ZHU, Q.; ZAKARIA, J.; KEOGH, E. Searching and mining trillions of time series subsequences under dynamic time warping. In: ACM. **SIGKDD International Conference on Knowledge Discovery and Data Mining**. Beijing, China, 2012. p. 262–270. Citations on pages [26](#), [38](#), [46](#), [47](#), [49](#), [53](#), [70](#), [94](#), [98](#), [99](#), [142](#) e [148](#).

RATANAMAHATANA, C. A.; KEOGH, E. Making time series classification more accurate using learned constraints. In: SIAM. **SIAM International Conference on Data Mining**. Lake Buena Vista, FL, USA, 2004. p. 11–22. Citations on pages [28](#) e [33](#).

_____. Three myths about dynamic time warping data mining. In: SIAM. **SIAM International Conference on Data Mining**. Newport Beach, CA, USA, 2005. p. 506–510. Citations on pages [26](#), [53](#), [54](#), [57](#), [62](#), [87](#), [118](#), [146](#) e [154](#).

REBBAPRAGADA, U.; PROTOPAPAS, P.; BRODLEY, C. E.; ALCOCK, C. Finding anomalous periodic time series. **Machine learning**, Springer, v. 74, n. 3, p. 281–313, 2009. Citations on pages [53](#) e [54](#).

REISS, A.; STRICKER, D. Introducing a new benchmarked dataset for activity monitoring. In: IEEE. **International Symposium on Wearable Computers**. Newcastle, UK, 2012. p. 108–109. Citation on page 111.

REN, Z.; FAN, C.; MING, Y. Music retrieval based on rhythm content and dynamic time warping method. In: IEEE. **IEEE International Conference on Signal Processing**. Hong Kong, China, 2016. p. 989–992. Citations on pages 43, 116 e 171.

RODRIGUEZ, A.; LAIO, A. Clustering by fast search and find of density peaks. **Science**, American Association for the Advancement of Science, v. 344, n. 6191, p. 1492–1496, 2014. Citation on page 41.

SAINI, I.; SINGH, D.; KHOSLA, A. Qrs detection using k-nearest neighbor algorithm (knn) and evaluation on standard ecg databases. **Journal of advanced research**, Elsevier, v. 4, n. 4, p. 331–344, 2013. Citation on page 53.

SAKOE, H.; CHIBA, S. Dynamic programming algorithm optimization for spoken word recognition. **IEEE Transactions on Acoustics, Speech and Signal Processing**, IEEE, v. 26, n. 1, p. 43–49, 1978. Citations on pages 26, 28, 33, 57, 76, 77, 97, 98 e 146.

SAKURAI, Y.; YOSHIKAWA, M.; FALOUTSOS, C. FTW: fast similarity search under the time warping distance. In: ACM. **ACM Symposium on Principles of Database Systems**. New York, NY, USA, 2005. p. 326–337. Citation on page 80.

SALVADOR, S.; CHAN, P. Toward accurate dynamic time warping in linear time and space. **Intelligent Data Analysis**, IOS Press, v. 11, n. 5, p. 561–580, 2007. Citations on pages 26, 34, 35, 76, 79, 122 e 129.

SAPP, C. **The Mazurka Project**. 2017. <<http://www.mazurka.org.uk>>. Accessed 26th Jul, 2017. Citation on page 131.

SCHÄFER, P. The boss is concerned with time series classification in the presence of noise. **Data Mining and Knowledge Discovery**, Springer Science & Business Media, v. 29, n. 6, p. 1505, 2015. Citation on page 39.

SCHÄFER, P.; LESER, U. Fast and accurate time series classification with weasel. **arXiv preprint arXiv:1701.07681**, 2017. Citation on page 39.

SERRA, J.; GÓMEZ, E.; HERRERA, P.; SERRA, X. Chroma binary similarity and local alignment applied to cover song identification. **IEEE Transactions on Audio, Speech, and Language Processing**, IEEE, v. 16, n. 6, p. 1138–1151, 2008. Citations on pages 123, 129, 131 e 132.

SERRA, J.; MÜLLER, M.; GROSCHE, P.; ARCOS, J. L. Unsupervised music structure annotation by time series structure features and segment similarity. **IEEE Transactions on Multimedia**, IEEE, v. 16, n. 5, p. 1229–1240, 2014. Citation on page 123.

SERRA, J.; SERRA, X.; ANDRZEJAK, R. G. Cross recurrence quantification for cover song identification. **New Journal of Physics**, IOP Publishing, v. 11, n. 9, p. 093017, 2009. Citation on page 124.

SHEN, Y.; CHEN, Y.; KEOGH, E.; JIN, H. Searching time series with invariance to large amounts of uniform scaling. In: IEEE. **IEEE International Conference on Data Engineering**. San Diego, CA, USA, 2017. p. 111–114. Citation on page 116.

SHOKOOHI-YEKTA, M.; HU, B.; JIN, H.; WANG, J.; KEOGH, E. Generalizing dtw to the multi-dimensional case requires an adaptive approach. **Data Mining and Knowledge Discovery**, Springer, v. 1, n. 31, p. 1–31, 2017. Citations on pages 110, 122 e 170.

SIGTIA, S.; DIXON, S. Improved music feature learning with deep neural networks. In: IEEE. **IEEE International Conference on Acoustics, Speech and Signal Processing**. Florence, Italy, 2014. p. 6959–6963. Citations on pages 131 e 170.

SILVA, D. F.; BATISTA, E. A. P. A.; KEOGH, E. **Prefix and Suffix Invariant DTW - Website**. 2016. <<https://sites.google.com/site/psidtw/>>. Accessed 26th Jul, 2017. Citations on pages 62 e 64.

SILVA, D. F.; BATISTA, E. A. P. A. B. **Speeding Up All-Pairwise Dynamic Time Warping Matrix Calculation - Website**. 2016. <<http://sites.labc.icmc.usp.br/prunedDTW/>>. Accessed 26th Jul, 2017. Citations on pages 85 e 87.

SILVA, D. F.; BATISTA, G. E. A. P. A. Speeding up all-pairwise dynamic time warping matrix calculation. In: SIAM. **SIAM International Conference on Data Mining**. Miami, FL, USA, 2016. p. 837–845. Citations on pages 46, 94, 102, 104, 119, 142 e 167.

SILVA, D. F.; BATISTA, G. E. A. P. A.; KEOGH, E. Prefix and suffix invariant dynamic time warping. In: IEEE. **IEEE International conference on Data Mining**. Barcelona, Spain, 2016. p. 1209–1214. Citations on pages 45, 120, 142, 147, 167, 169 e 170.

SILVA, D. F.; GIUSTI, R.; KEOGH, E.; BATISTA, G. E. A. P. A. **UCR-USP Suite website**. 2016. <<https://sites.google.com/view/ucruspsuite>>. Accessed 26th Jul, 2017. Citation on page 110.

SILVA, D. F.; PAPADOPOULOS, H.; BATISTA, G. E. A. P. A.; ELLIS, D. P. W. A video compression-based approach to measure music structural similarity. In: ISMIR. **International Society for Music Information Retrieval Conference**. Curitiba, PR, Brazil, 2013. p. 95–100. Citation on page 132.

SILVA, D. F.; ROSSI, R. G.; REZENDE, S. O.; BATISTA, G. E. A. P. A. Music classification by transductive learning using bipartite heterogeneous networks. In: ISMIR. **International Society for Music Information Retrieval Conference**. Taipei, Taiwan, 2014. p. 113–118. Citations on pages 40, 165 e 168.

SILVA, D. F.; SOUZA, V. M. A.; BATISTA, G. E. A. P. A. Music shapelets for fast cover song recognition. In: ISMIR. **International Society for Music Information Retrieval Conference**. Malaga, Spain, 2015. p. 441–447. Citations on pages 129, 131, 132, 165 e 167.

SILVA, D. F.; YEH, C.-C. M.; BATISTA, G. E. A. P. A.; KEOGH, E. **Supporting website for the SiMPle-Fast paper**. <<https://sites.google.com/view/simple-fast>>. Accessed 26th Jul, 2017. Citations on pages 139 e 140.

_____. SiMPle: assessing music similarity using subsequences joins. In: ISMIR. **International Society for Music Information Retrieval Conference**. New York, NY, USA, 2016. p. 23–29. Citations on pages 48, 126, 134, 167, 169 e 171.

SOUZA, V. M. A.; SILVA, D. F.; BATISTA, G. E. A. P. A. Extracting texture features for time series classification. In: IEEE. **International Conference on Pattern Recognition**. Prague, Czech Republic, 2014. p. 1425–1430. Citations on pages 166 e 168.

SOUZA, V. M. A.; SILVA, D. F.; BATISTA, G. E. A. P. A.; GAMA, J. Classification of evolving data streams with infinitely delayed labels. In: IEEE. **IEEE International Conference on Machine Learning and Applications**. Miami, FL, USA, 2015. p. 214–219. Citations on pages [166](#) e [168](#).

SOUZA, V. M. A.; SILVA, D. F.; GAMA, J.; BATISTA, G. E. A. P. A. Data stream classification guided by clustering on nonstationary environments and extreme verification latency. In: SIAM. **SIAM International Conference on Data Mining**. Vancouver, Canada, 2015. p. 873–881. Citations on pages [166](#) e [168](#).

SPIEGEL, S.; JAIN, B.-J.; ALBAYRAK, S. Fast time series classification under lucky time warping distance. In: ACM. **ACM Symposium on Applied Computing**. Gyeongju, Korea, 2014. p. 71–78. Citations on pages [26](#), [34](#), [76](#) e [79](#).

STEFAN, A.; ATHITSOS, V.; DAS, G. The move-split-merge metric for time series. **IEEE transactions on Knowledge and Data Engineering**, IEEE, v. 25, n. 6, p. 1425–1438, 2013. Citation on page [120](#).

STEIN, M.; JANETZKO, H.; SEEBACHER, D.; JÄGER, A.; NAGEL, M.; HÖLSCH, J.; KOSUB, S.; SCHRECK, T.; KEIM, D. A.; GROSSNIKLAUS, M. How to make sense of team sport data: From acquisition to data modeling and research aspects. **Data**, Multidisciplinary Digital Publishing Institute, v. 2, n. 1, p. 2, 2017. Citation on page [170](#).

STEINWART, I.; CHRISTMANN, A. **Support vector machines**. New York, NY, USA: Springer New York, 2008. Citation on page [39](#).

SUHRBIER, A.; HERINGER, R.; WALTHER, T.; MALBERG, H.; WESSEL, N. Comparison of three methods for beat-to-beat-interval extraction from continuous blood pressure and electrocardiogram with respect to heart rate variability analysis. **Biomedizinische Technik**, v. 51, n. 2, p. 70–76, 2006. Citation on page [53](#).

SWAN, M. Sensor mania! the internet of things, wearable computing, objective metrics, and the quantified self 2.0. **Journal of Sensor and Actuator Networks**, v. 1, n. 3, p. 217–253, 2012. Citation on page [51](#).

TABORRI, J.; PALERMO, E.; ROSSI, S.; CAPPÀ, P. Gait partitioning methods: A systematic review. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 16, n. 1, p. 66, 2016. Citation on page [53](#).

TANG, H.; LIAO, S. S. Discovering original motifs with different lengths from time series. **Knowledge Based Systems**, Elsevier, v. 21, n. 7, p. 666–671, 2008. Citations on pages [142](#) e [147](#).

THELWALL, M. Sentiment analysis and time series with twitter. **Twitter and Society**. Peter Lang Publishing, p. 83–96, 2014. Citation on page [44](#).

TOOHEY, K.; DUCKHAM, M. Trajectory similarity measures. **SIGSPATIAL Special**, ACM, v. 7, n. 1, p. 43–50, 2015. Citation on page [170](#).

TORKAMANI, S.; LOHWEG, V. Survey on time series motif discovery. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, Wiley Online Library, v. 7, n. 2, 2017. Citations on pages [25](#) e [49](#).

TORMENE, P.; GIORGINO, T.; QUAGLINI, S.; STEFANELLI, M. Matching incomplete time series with dynamic time warping: an algorithm and an application to post-stroke rehabilitation. **Artificial intelligence in medicine**, Elsevier, v. 45, n. 1, p. 11–34, 2009. Citation on page 57.

TRUONG, C. D.; ANH, D. T. A fast method for motif discovery in large time series database under dynamic time warping. In: **International Conference On Knowledge And Systems Engineering**. Ho Chi Minh, Vietnam: KSE, 2015. p. 155–167. Citation on page 142.

TSAI, W.-H.; YU, H.-M.; WANG, H.-M. Using the similarity of main melodies to identify cover versions of popular songs for music document retrieval. **Journal of Information Science and Engineering**, v. 24, n. 6, p. 1669–1687, 2008. Citation on page 129.

UENO, K.; XI, X.; KEOGH, E.; LEE, D.-J. Anytime classification using the nearest neighbor algorithm with applications to stream mining. In: IEEE. **IEEE International Conference on Data Mining**. Hong Kong, China, 2006. p. 623–632. Citations on pages 54, 62, 64 e 166.

ULANOVA, L.; BEGUM, N.; KEOGH, E. Scalable clustering of time series with u-shapelets. In: SIAM. **SIAM International Conference on Data Mining**. Vancouver, Canada, 2015. p. 900–908. Citations on pages 25 e 78.

VAIL, D.; VELOSO, M. Learning from accelerometer data on a legged robot. In: **IFAC/EURON symposium on intelligent autonomous vehicles**. Lisbon, Portugal: Elsevier, 2004. Citation on page 65.

VLACHOS, M.; HADJIELEFTHERIOU, M.; GUNOPULOS, D.; KEOGH, E. Indexing multi-dimensional time-series with support for multiple distance measures. In: ACM. **SIGKDD International Conference on Knowledge Discovery and Data Mining**. Washington, DC, USA, 2003. p. 216–225. Citation on page 31.

_____. Indexing multidimensional time-series. **The VLDB Journal**, Springer-Verlag New York, Inc., v. 15, n. 1, p. 1–20, 2006. Citation on page 121.

WALKER, J. S. **Fast fourier transforms**. Boca Raton, FL, USA: CRC press, 1996. Citation on page 42.

WANG, X.; MUEEN, A.; DING, H.; TRAJCEVSKI, G.; SCHEUERMANN, P.; KEOGH, E. Experimental comparison of representation methods and distance measures for time series data. **Data Mining and Knowledge Discovery**, Springer, v. 26, n. 2, p. 275–309, 2013. Citations on pages 25, 54, 56, 59, 62, 75, 94, 96, 98, 120, 142 e 146.

WATTENBERG, M. **The shape of song**. <<http://www.turbulence.org/Works/song/mono.html>>. Accessed 26th Jul, 2017. Citation on page 138.

_____. Arc diagrams: Visualizing structure in strings. In: IEEE. **IEEE Symposium on Information Visualization**. Boston, MA, USA, 2002. p. 110–116. Citation on page 138.

WU, H.-H.; BELLO, J. P. Audio-based music visualization for music structure analysis. In: SMC. **Sound and Music Computing Conference**. Barcelona, Spain, 2010. Citations on pages 123 e 138.

XU, R.; WUNSCH, D. **Clustering**. Piscataway, NJ, USA: John Wiley & Sons, 2008. Citations on pages 58 e 102.

YABE, T.; TANAKA, K. Similarity retrieval of human motion as multi-stream time series data. In: IEEE. **Database Applications in Non-Traditional Environments**. Kyoto, Japan, 1999. p. 279–286. Citation on page [155](#).

YE, L.; KEOGH, E. Time series shapelets: a new primitive for data mining. In: ACM. **SIGKDD International Conference on Knowledge Discovery and Data Mining**. Paris, France, 2009. p. 947–956. Citation on page [165](#).

YEH, C.-C. M.; ZHU, Y.; ULANOVA, L.; BEGUM, N.; DING, Y.; DAU, H. A.; SILVA, D. F.; MUEEN, A.; KEOGH, E. Matrix Profile I: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In: IEEE. **IEEE International Conference on Data Mining**. Barcelona, Spain, 2016. p. 1317–1322. Citations on pages [42](#), [48](#), [128](#), [142](#), [148](#), [165](#), [167](#) e [169](#).

YEH, C.-C. M.; ZHU, Y.; ULANOVA, L.; BEGUM, N.; DING, Y.; DAU, H. A.; ZIMMERMAN, Z.; SILVA, D. F.; MUEEN, A.; KEOGH, E. Time series joins, motifs, discords and shapelets: a unifying view that exploits the matrix profile. **Data Mining and Knowledge Discovery**, Springer, p. 1–41, 2017. Citations on pages [42](#), [165](#) e [167](#).

ZHENG, Y. Trajectory data mining: an overview. **ACM Transactions on Intelligent Systems and Technology**, ACM, v. 6, n. 3, p. 29, 2015. Citation on page [170](#).

ZHU, Q.; BATISTA, G. E. A. P. A.; RAKTHANMANON, T.; KEOGH, E. A novel approximation to dynamic time warping allows anytime clustering of massive time series datasets. In: SIAM. **SIAM International Conference on Data Mining**. Anaheim, CA, USA, 2012. p. 999–1010. Citations on pages [26](#), [35](#), [76](#), [78](#) e [79](#).

ZHU, Y.; ZIMMERMAN, Z.; SENOBARI, N. S.; YEH, C.-C. M.; FUNNING, G.; MUEEN, A.; BRISK, P.; KEOGH, E. Matrix Profile II: Exploiting a novel algorithm and gpus to break the one hundred million barrier for time series motifs and joins. In: IEEE. **IEEE International Conference on Data Mining**. Barcelona, Spain, 2016. p. 739–748. Citations on pages [42](#), [48](#), [128](#) e [154](#).

ZILBERSTEIN, S. Using anytime algorithms in intelligent systems. **AI magazine**, v. 17, n. 3, p. 73, 1996. Citation on page [35](#).

ZUNIC, J.; ROSIN, P. L.; KOPANJA, L. On the orientability of shapes. **IEEE Transactions on Image Processing**, IEEE, v. 15, n. 11, p. 3478–3487, 2006. Citation on page [31](#).