Università della Calabria

Dipartimento di Matematica e Informatica

Dottorato di Ricerca in Matematica e Informatica

XXX ciclo

Tesi di Dottorato

Generalizing Identity-Based String Similarity Metrics: Theory and Applications

Settore Disciplinare INF/01 – INFORMATICA

Coordinatore: Ch.mo Prof. Nicola Leone

Supervisore: Prof. Giorgio Terracina

Dottorando: Dott. Francesco Cauteruccio

A mia madre.

Acknowledgments

First of all, I would like to thank my supervisor Professor Giorgio Terracina. He has been an inspired figure for me during the whole course of this “adventure”. His advices and suggestions helped me become the person I actually am. His knowledge inspired me and I can just consider myself lucky and honored for his consideration towards myself.

Among all of the people I’ve worked with, I would like to thank Professor Domenico Ursino. His patient guidance in all of our works has been of great inspiration for me. Also, I would like to thank Doctor Dominique Sappey-Marinier who embraced my research visiting period in Lyon. Furthermore, I would also to thank Professor Nicola Leone, Coordinator of the Ph.D. programme in Mathematics and Computer Science.

~

Ci siamo. Quando queste parole abiteranno una città di carta, questa tesi sarà indelebilmente terminata e degli occhi, adesso, staranno navigando queste parole. Qui, i miei ringraziamenti informali. Per iniziare, ringrazio la mia famiglia, incommensurabile costante della mia vita, matrice della mia esistenza. Ringrazio Francesco, che conosco da piccino e che rappresenta per me fonte di grande ammirazione. Ringrazio ancora una volta lei, che mi ha ascoltato senza che io dicessi alcunché. Ringrazio Ferdinando, il mio migliore amico e fidato (scarso) support, che è sempre qui vicino anche se a più di duemila chilometri di distanza. Ringrazio Claudia, la mia migliore amica, che è al mio fianco anche quando io non ci sono. Ringrazio Jennifer, una delle persone più umili e intelligenti che io conosca, e soprattutto mia cara amica. Ringrazio i miei amici di sempre Angelo, Antonio e Corrado, che sono sempre qui e che sanno trasformare un insieme di semplici passaggi in un’epocale odissea. Ringrazio Claudio, che oltre ad essere un collega è soprattutto un sacro amico, per il quale affetto e stima si sommano. Ringrazio, fra tutti, Aldo, Carmelo, Antonella, Marianna e Roberta, e tutte le altre persone che probabilmente, in questa rocambolesca valanga di emozioni, ho dimenticato. Infine, senza fine alcuna, ringrazio Ilaria, per avermi insegnato come anch’io possa essere un *mostrum*.

“Chi non dà nulla non ha nulla. La disgrazia più grande non è non essere amati, ma non amare.”

— Albert Camus, “Taccuini 1935–1959”

Abstract

Strings play a fundamental role in computer science. Data is codified into strings, and by interpreting them information can be derived. Given a set of strings, few interesting questions arise, such as *"are these strings related?"*, and *"if they are related, how can we measure this relatedness?"*. The definition of a degree of similarity (or correlation) between strings is strongly important. Different definitions of similarity between strings already exist in literature and they steam from the concept of *metric* in mathematics. One of the most famous and well-known string similarity metric is the *edit distance*, which measures the minimum number of *edit operations* required to transform one string into another one. However, in the definition of the similarity between two strings, one important natural assumption is made: identical symbols among strings represent identical information, whereas different symbols introduce some form of differentiation. This last assumption results to be extremely reductive. In fact, there are cases in which symbol identity seems to be not enough, and even if there are no common symbols between two strings, it could happen that they represent similar information. Moreover, there are cases in which a one-to-one mapping between symbols is not enough, thus a *many-to-many* mapping is needed. The necessity of a suitable metric capable of capturing *hidden* correlations between strings emerges and this metric should take into account that *different* symbols may express *similar* concepts.

This thesis aims to provide a contribution in this setting. Initially, we present a framework that generalizes most of the existing string metrics based on symbol identity, making them suitable for application scenarios which involve strings defined on heterogeneous alphabets. We formally define the Multi-Parameterized Edit Distance, a generalization of the edit distance with the support of our framework, and we discuss its computational issues.

Then, we present various heuristics designed, implemented and tested out, in order to approach computational issues of the generalization: we start with a survey on heuristics to acquire a global view of the problem, then we select, discuss and test three of them in detail.

In the last part, we discuss several application contexts which have been studied in this thesis. These scenarios span from engineering to biomedical informatics. In particular, they concentrate on Wireless Sensors Area Networks, White Matter Fiber-Bundles analysis and Electroencephalogram analysis.

Finally, at the end of the thesis, we draw our conclusions and highlight future work.

Sommario

Le stringhe giocano un ruolo fondamentale in informatica: codificando i dati, la loro interpretazione permette di derivare informazione. Dato un insieme di stringhe, alcune interessanti domande emergono: *"queste stringhe sono correlate?"*, e se lo sono, *"possiamo misura la loro correlazione?"*. La definizione di un grado di similarità tra stringhe risulta essere fortemente importante. Varie definizioni di similarità tra stringhe sono state definite nella letteratura, derivanti dal concetto di *metrica* in matematica. Una delle più famose metriche di similarità tra stringhe è la *edit distance*, definita come il numero minimo di *edit operation* necessarie a trasformare una stringa in un'altra. Tuttavia, le varie definizioni presentano un'assunzione chiave: simboli uguali tra le stringhe rappresentano la stessa identica informazione, mentre simboli diversi introducono una qualche differenza. Questa assunzione risulta essere estremamente riduttiva: esistono casi in cui l'identità tra simboli sembra non essere sufficiente a definire una similarità, e nel caso in cui non ci siano simboli in comune tra due stringhe, si può verificare che simboli diversi rappresentino la stessa informazione. Inoltre, in alcuni casi una mappatura *one-to-one* tra i simboli risulta inefficace, quindi si necessita una mappatura *many-to-many*. La necessità di avere una metrica di similarità tra stringhe che sia in grado di catturare *correlazioni nascoste* tra le stringhe emerge, ove il concetto chiave è rappresentato dal considerare che simboli *differenti* possono esprimere concetti *simili*.

Lo scopo di questa tesi è di contribuire in questo scenario. In primis, un framework che generalizza la maggior parte delle metriche di similarità tra stringhe (basate sull'identità tra simboli) viene presentato, idoneo a scenari di applicazione in cui sono presenti stringhe definite su alfabeti eterogenei. La Multi-Parameterized Edit Distance (una generalizzazione della edit distance con il supporto del framework) viene definita formalmente e studiata dal punto di vista della complessità computazionale.

In seguito, differenti euristiche, definite, implementate e testate, vengono presentate, in modo da approcciarsi alle difficoltà computazionali presenti. Varie euristiche sono presentate e tre di esse sono studiate, discusse e testate in dettaglio.

Alcuni contesti di applicazione, studiati in questa tesi, sono quindi discussi, spaziando dal settore ingegneristico a quello informatico biomedico: anomaly detection nelle Wireless Sensors Area Network, analisi dei White Matter Fiber-Bundles e analisi degli Elettroencefalogrammi. Le conclusioni e una panoramica dei lavori futuri chiudono la tesi.

Contents

Abstract v

Sommario vi

Contents vii

List of Figures xi

List of Tables xv

1 Introduction 19

2 Background and Problem Definition 25

2.1 Introduction 25

2.2 The fundamental concept of string 25

2.3 Similarity, correlation and distance 26

2.4 The real problem 28

2.5 Related work 29

3 Generalizing identity-based string comparison metrics 33

3.1 Introduction 33

3.2 Preliminaries 33

3.3 The framework 36

3.4 Generalization of notable string similarity metrics 36

3.4.1 Edit Distance 37

3.4.2 Affine Gap Distance 37

3.4.3 Smith-Waterman Distance 38

3.4.4 Jaro Distance Metric 38

3.4.5 Atomic Strings 38

3.4.6 WHIRL 39

3.4.7 Q-grams with *tf.idf* 39

3.4.8 Parameterized pattern matching 40

4 Multi-Parameterized Edit Distance **41**

4.1 Introduction 41

4.2 Definitions 41

4.3 Examples 43

4.4 Computational issues: complexity 44

4.4.1 NP-Hardness 44

4.4.2 A lower bound L 47

5 Attacking the giant: heuristic approaches **51**

5.1 Introduction 51

5.2 A global view on heuristics 52

5.2.1 HeuristicLab 53

5.2.2 Comparison of different heuristics 58

5.3 Survey on local search heuristics 71

5.3.1 Hill Climbing 71

5.3.2 Simulated Annealing 75

5.3.3 Experimental Analysis 75

5.4 Survey on evolution strategies 82

5.4.1 Components of an Evolutionary Algorithm 82

5.4.2 Evolution Strategy 86

5.4.3 Fundamental Components of Evolution Strategy for MPED 88

5.4.4 Evolution Strategy Implementations 89

5.4.5 Experiments 91

6 Applications of the Multi-Parameterized Edit Distance **121**

6.1 Introduction 121

6.2 MPED for Wireless Sensor Area Networks 122

6.2.1 Background . . . 122
6.2.2 Case study . . . 124
6.2.3 Experiments and discussion . . . 125
6.2.4 Hidden correlation for different positioning of the sensors nodes . . . 126
6.2.5 Robustness of the measure . . . 128
6.2.6 Sensitivity to sensor faults . . . 128
6.3 MPED for White Matter Fiber-Bundles . . . 130
6.3.1 Background . . . 130
6.3.2 Extract and characterize White Matter Fiber-Bundles . . . 131
6.3.3 Integration of spatial information . . . 150
6.4 MPED for Electroencephalograms . . . 157
6.4.1 Background . . . 157
6.4.2 Core ingredients . . . 159
6.4.3 The proposed approach . . . 162
6.4.4 Investigating neurological disorders . . . 167
6.4.5 Discussion . . . 177

7 Conclusion **181**

Bibliography **183**

List of Figures

5-1 Applying a genetic algorithm for a vehicle routing problem. A video tour of HL and further information can be found at https://dev.heuristiclab.com/trac.fcgi/ 54

5-2 Runs with alphabets cardinality 16 and added correlation 0 62

5-3 Runs with alphabets cardinality 16 and added correlation 0.5 63

5-4 Runs with alphabets cardinality 16 and added correlation 1 64

5-5 Runs with alphabets cardinality 12 and added correlation 0 65

5-6 Runs with alphabets cardinality 12 and added correlation 0.5 66

5-7 Runs with alphabets cardinality 12 and added correlation 1 67

5-8 Runs with alphabets cardinality 8 and added correlation 0 68

5-9 Runs with alphabets cardinality 8 and added correlation 0.5 69

5-10 Runs with alphabets cardinality 8 and added correlation 1 70

5-11 Examples of $\langle\pi_1, \pi_2\rangle$-matching schemas for $\pi_1 = 3$ and $\pi_2 = 2$ 72

5-12 Execution time (in seconds) of HC, SA and EX against $|\Pi|$. The four graphs, from top to bottom and from left to right, indicate the results for $len(s) =$ 100, 200, 350 and 500, respectively. 79

5-13 Execution time (in seconds) of HC, SA and EX against $len(s)$. The six graphs, from top to bottom and from left to right, indicate the result for $|\Pi| \in \{5..10\}$. 80

5-14 Execution time (in seconds) of HC against $len(s)$ for several values of $|\Pi|$. . 81

5-15 Runtime comparison of ES and HC with $|\Pi| = 14, len(s) = 2000, \pi_1 = \pi_2 = 1$ 97

5-16 Runtime comparison of ES and HC with $|\Pi| = 16, len(s) = 2000, \pi_1 = \pi_2 = 1$ 97

5-17 Runtime comparison of ES and HC with $|\Pi| = 18, len(s) = 2000, \pi_1 = \pi_2 = 1$ 98

5-18 Runtime comparison of ES and HC with $|\Pi| = 20, len(s) = 2000, \pi_1 = \pi_2 = 1$ 98

5-19 Runtime comparison of ES and HC with $|\Pi| = 16, len(s) = 2000, \pi_1 = \pi_2 = 2$ 99

5-20 Runtime comparison of ES and HC with $|\Pi| = 16, len(s) = 2000, \pi_1 = \pi_2 = 3$ 99

5-21 Runtime comparison of ES and HC with $|\Pi| = 16, len(s) = 2000, \pi_1 = \pi_2 = 4$ 100

5-22 Runtime comparison of ES and HC with $|\Pi| = 16, len(s) = 2000, \pi_1 = \pi_2 = 5$ 100

5-23 Comparison of mutation operators with $|\Pi| = 20, len(s) = 1000, \pi_1 = \pi_2 = \{1, 2, 3\}$ . . . 105

5-24 Comparison of mutation operators with $|\Pi| = 20, len(s) = 1000, \pi_1 = \pi_2 = \{4, 5, 6\}$ . . . 106

5-25 Comparison of mutation operators with $|\Pi| = 20, len(s) = 1000, \pi_1 = \pi_2 = \{7, 8, 9\}$ . . . 107

5-26 Comparison of mutation operators with $|\Pi| = 20, len(s) = 1000, \pi_1 = \pi_2 = 10$ 108

5-27 Comparison of ES variants with $|\Pi| = 20, len(s) = 1000, \pi_1 = \pi_2 = \{1, 2, 3\}$ 110

5-28 Comparison of ES variants with $|\Pi| = 20, len(s) = 1000, \pi_1 = \pi_2 = \{4, 5, 6\}$ 111

5-29 Comparison of ES variants with $|\Pi| = 20, len(s) = 1000, \pi_1 = \pi_2 = \{7, 8, 9\}$ 112

5-30 Comparison of ES variants with $|\Pi| = 20, len(s) = 1000, \pi_1 = \pi_2 = 10$ . . . 113

5-31 Probability distribution for a particular MPED value with $|\Pi| = 20, len(s) = 1000, \pi_1 = \pi_2 = 1$ . . . 116

5-32 Probability distribution for a particular MPED value with $|\Pi| = 20, len(s) = 1000, \pi_1 = \pi_2 = \{2, 3, 4\}$ . . . 117

5-33 Probability distribution for a particular MPED value with $|\Pi| = 20, len(s) = 1000, \pi_1 = \pi_2 = \{5, 6, 7\}$ . . . 118

5-34 Probability distribution for a particular MPED value with $|\Pi| = 20, len(s) = 1000, \pi_1 = \pi_2 = \{8, 9, 10\}$ . . . 119

6-1 Wireless sensor nodes deployed at DIMES. . . . 125

6-2 A BMF Network. . . . 126

6-3 Plot of (T)emperature and (L)ight collected from nodes a,b,c,d . . . 127

6-4 Sensitivity to sensor faults, Node *(d)* . . . 129

6-5 The virtual phantom used for our experimental campaign . . . 141

6-6 The clusters generated by our approach with the adoption of k-means as clustering algorithm . . . 143

6-7 The clusters generated by QuickBundles . . . 143

6-8 The 17 bundles identified in the diffusion MR phantom adopted in our test 145

6-9 **a.** Approximate shape of Corpus Callosum (CC) and its axis of symmetry (black dotted line) drew by the operator; **b.** Extracted forcep minor of CC fibers (green) 149

6-10 **a.** Approximate shape of Cortico-Spinal Tract (CST) and its axis of symmetry (black dotted line) drew by the operator; **b.** Extracted right CST fibers (green) 150

6-11 The two phantoms used in our experimental campaign 153

6-12 Variation of the four performance measures against the threshold Th for each model of the phantom of Figure 6-11(a) 154

6-13 Distribution of the edge weights and colored network of a control subject, a patient with MCI and a patient with AD 165

6-14 Partitioning of an EEG into segments with PSWCs and without PSWCs . . 168

6-15 Colored Networks $\mathcal{N}_\pi$ and $\overline{\mathcal{N}_\pi}$ for a patient with CJD 169

6-16 Connection coefficient for the network $\mathcal{N}^{blk}$ 172

6-17 Connection coefficient for the network $\mathcal{N}^{wht}$ 172

6-18 Density for the network $\mathcal{N}^{blk}$ 173

6-19 Density for the network $\mathcal{N}^{wht}$ 173

6-20 Clustering coefficient for the network $\mathcal{N}^{blk}$ 174

6-21 Clustering coefficient for the network $\mathcal{N}^{wht}$ 174

6-22 The networks $\mathcal{N}_{0_\pi}$ and $\mathcal{N}_{1_\pi}$ for two patients with MCI at both t_0 and t_1 . . 176

6-23 The networks $\mathcal{N}_{0_\pi}$ and $\mathcal{N}_{1_\pi}$ for two patients with MCI at t_0, who converted to AD at t_1 178

6-24 Results of the application of the approach of [80] to the patients of Figures 6-22 and 6-23 179

List of Tables

2.1 Few examples of strings. 26

5.1 Parameters for Local Search heuristic 55

5.2 Parameters for Simulated Annealing heuristic 55

5.3 Parameters for Genetic Algorithm heuristic 56

5.4 Parameters for Offspring Selection Genetic Algorithm heuristic 57

5.5 Parameters for Genetic Algorithm with ALPS heuristic 57

5.6 Parameters for Evolution Strategy algorithm 58

5.7 Parameters for Offspring Selection Evolution Strategy algorithm 58

5.8 Comparative benchmark parameters for HeuristicLab heuristics comparison. 58

5.9 Lowest obtained MPED value among all runs and heuristics for HeuristicLab heuristics comparison. 60

5.10 HL heuristics comparison, alphabet cardinality 16 and added correlation 0 . 62

5.11 HL heuristics comparison, alphabet cardinality 16 and added correlation 0.5 63

5.12 HL heuristics comparison, alphabet cardinality 16 and added correlation 1 . 64

5.13 HL heuristics comparison, alphabet cardinality 12 and added correlation 0 . 65

5.14 HL heuristics comparison, alphabet cardinality 12 and added correlation 0.5 66

5.15 HL heuristics comparison, alphabet cardinality 12 and added correlation 1 . 67

5.16 HL heuristics comparison, alphabet cardinality 8 and added correlation 0 . 68

5.17 HL heuristics comparison, alphabet cardinality 8 and added correlation 0.5 69

5.18 HL heuristics comparison, alphabet cardinality 8 and added correlation 1 . 70

5.19 Values of P_{HC} (left value) and P_{SA} (right value) for different values of π, $|\Pi|$ and $len(s)$. N/A represents an instance for which the exact value cannot be computed in a feasible time. 78

5.20 Parameters and values for each configuration of $(\mu+\lambda)-ES$ vs an exhaustive approach. 93
5.21 Results for $(\mu+\lambda)-ES$ with configuration *strong*. 94
5.22 Results for $(\mu+\lambda)-ES$ with configuration *medium*. 94
5.23 Results for $(\mu+\lambda)-ES$ with configuration *light*. 95
5.24 Selected values for ES parameters. 96
5.25 Obtained precision of $(\mu+\lambda)-ES$ w.r.t HC. 102
5.26 Runtime analysis of $(\mu+\lambda)-ES$ w.r.t HC. 103
5.27 Selected values for ES parameters within mutation operators tests. 104
5.28 Estimated precision P_M for each mutation operator for different values of π_i. 109
5.29 Estimated precision P_V for each ES variant and different values of π. 114
5.30 Configurations of $(\mu+\lambda)-ES$ for probability distribution test. 115

6.1 MPED and Std Correlation 128
6.2 Day span 128
6.3 Results for Cluster 1 142
6.4 Results for Cluster 2 142
6.5 Results for Cluster 3 142
6.6 Results for Cluster 4 142
6.7 Results obtained by applying MPED$_{SB}$ and the edit distance on the 17 models into consideration 146
6.8 Comparison of the average Precision, Recall, F-measure and Overall obtained by applying MPED$_{SB}$ and the edit distance 146
6.9 Performance values obtained by our approach and QB when applied on the phantom of Figure 6-11(a) 155
6.10 Performance values obtained by our approach and QB when applied on the phantom of Figure 6-11(b) 155
6.11 Quantitative measures representing the colored networks of Figure 6-13 . . 166
6.12 Table produced by a neurologist regarding starting and ending time-slots for each seizure of the EEG into consideration 171
6.13 Quantitative measures representing the networks $\mathcal{N}_{0_\pi}$ and $\mathcal{N}_{1_\pi}$ for the two patients considered in Figure 6-22 177

6.14 Quantitative measures representing the networks $\mathcal{N}_{0_\pi}$ and $\mathcal{N}_{1_\pi}$ for the two patients considered in Figure 6-23 177

6.15 Quantitative measures representing the results shown in Figure 6-24 179

6.16 Summary of Section 6.4.4 180

Chapter 1

Introduction

Strings play a fundamental role in computer science. Data is codified into strings by using various techniques and, by interpreting them, information can be derived. Almost everything in the digital world can be represented as strings: time series, access logs, astronomical data, this thesis, this exact phrase, everything that can be actually accessed. Even in the non-digital world, strings are everywhere: tram time schedules, road addresses, the plate of your car, the content of the book on your left, etc. The academic literature has been researching for many years over strings, making a fundamental theory that plays a crucial role in theoretical computer science. Information is the nourishment of computer science and with strings we encode and represent it.

From the point of view of the human society, experience strictly correlates to evolution. When we find ourselves in a novel situation, our brain automatically exploits our experience in order to understand whether we already encountered something *similar* to it or which is *correlated* to the situation itself. Terms like similarity and correlation are synonyms and represent very important concepts. These concepts also apply to computer science and to strings especially. Given a set of strings, few interesting questions arise, such as *"are these strings related?"*, *"if they are related, how can we measure this relatedness?"* and *"can we define some sort of measure of (dis)similarity between strings?"*. The definition of a degree of correlation or similarity between strings is strongly important: it gives us the possibility to extrapolate interesting properties, which can be used in various contexts. A straightforward example could be that of clustering a set of strings, determining whether two or more strings should be put "in the same cluster", being sufficiently similar to each other.

The academic community is very active on this aspect, thus different definitions of similarity/correlation between strings exist. Eventually, they steam from the concept of *metric* (or *distance function*) in mathematics, which is a function that defines a *distance* between each pair of elements of a set. In this case, distance is used to quantify the similarity between two strings. As an example, a famous and well-know metric in computer science is the *Hamming distance*, which measures the minimum number of substitutions required to change one string into the other. A large number of metrics exist and they are used in very heterogeneous contexts, such as information retrieval, machine learning, computational biology, etc.

Among these metrics, a particular one is the so called *edit distance*. It measures the minimum number of *edit operations* required to transform one string into another one. Edit operations can be of different kind, most common operations are deletions, insertions and substitutions of symbols. In this case, whether we try to define the similarity between two strings, one important natural assumption is made: identical symbols among strings represent identical information, whereas different symbols introduce, in a way or another, some form of differentiation. This last assumption results to be extremely reductive. In fact, there are cases in which symbol identity seems to be not enough, and even if there are no common symbols between two strings, it could happen that they represent similar information.

Let us consider two strings generated by heterogeneous data streams, derived from two different ways of measuring the same reality; think, for instance, of two sensors one measuring light and the other measuring temperature. The values, the scales and the meaning of the two sensors may be very different, but, if sensors are near to a fire, light can be influenced by temperature, and vice versa. To properly monitor this last event, we should, then, be able to understand the correlations between these two heterogeneous measurements. In another, extreme, case, suppose one of the streams has been deliberately manipulated in such a way as to appear dissimilar from the other (think for example to code cloning techniques [67]). In this case, it can be of outmost importance being able to detect their correlation. But how can we detect the similarity between such streams?

When dealing with heterogeneous data streams, generated sequences may come from very different contexts and may be represented with different symbols/metrics. Furthermore, we need to take in account that providing the flexibility of mapping some symbol of the first stream into more than one symbol of the second stream (and vice versa) would be extremely useful, thus *many-to-many* mappings between symbols need to be considered. This is also useful, for example, to accomodate different discretization metrics of numerical sequences. Obviously, the best mapping between symbols, i.e., the one which gives the minimum distance or which discovers the optimal correlation, is not know *a priori*, thus searching it becomes part of the problem.

The necessity of a suitable metric capable of capturing *hidden* correlations between strings emerges. This metric should take into account that *different* symbols in the involved strings may express *similar* concepts. An ideal naïve solution would be that of considering *all* of the possible correlations between symbols. However, this approach is infeasible; worse, it reduces to an intractable problem.

For a better understanding of this problem, which is discussed in detail in Chapter 2, let $s_1 = \texttt{AAABCCDCAA}$ and $s_2 = \texttt{EEFGHGGFHH}$ be two strings. They obviously have no symbols in common, thus resulting in being two strings completely different. However, suppose a hidden correlation exists, which matches symbols $\{\texttt{A,B}\}$ with $\{\texttt{E,H}\}$, and symbols $\{\texttt{C,D}\}$ with $\{\texttt{F,G}\}$. Thus, the following alignment is possible:

$$\begin{array}{lcl} s_1 : \texttt{AAABCCDDCAA} & \rightarrow & \texttt{AAABCCDDCAA} \\ s_2 : \texttt{EEFGHGGFHH} & \rightarrow & \texttt{-EEFGHGGFHH} \\ & & \texttt{ }\star\star\texttt{ }\star\texttt{ }\star\star\star\star\star \end{array}$$

which states that the second string could be obtained from the first one with just 3 (parameterized) edit operations, thus showing a signicant, not obvious, correlation between the two strings (see Chapter 4 for all details). Furthermore, different mappings could exists that might give a lower number of edit operations, but enumerating them all is in fact infeasible.

There are different contexts in which such a metric can be effectively used in order to approach particular problems. For instance, the engineering world is actually interested in the design, development and monitoring of wireless sensors area networks, whose complexity is constantly growing, thanks to the Internet of Things. Here, sensors transmit heterogeneous data, i.e., they are devices producing different kinds of signals. Challenges such as anomaly detection in this context would benefit from a metric which is capable of discovering hidden correlations between heterogeneous series of data. And shifting away from the engineering world, the biomedical context offers various scenarios in which such a metric would be useful. For example, White Matter Fiber-Bundle analysis concentrates on the investigation of brain, in which the capability of extracting, visualizing and analyzing White Matter fibers play a key role. This analysis is important because it may provide significant insights in brain functions and anomalies, in order to better understand and predict different neurodegenerative pathologies, such as multiple sclerosis. Here, providing a suitable string representation of the fibers, processes that were time consuming and error prone for a large cohort of patients would be more accessible.

In this thesis, we provide a contribution in this setting. We present a framework that generalizes most of the existing string metrics based on symbol identity making them suitable for application scenarios where involved strings could be based on heterogeneous alphabets. This way, our approach paves the way to the adoption of string metrics in all those contexts in which involved strings come from different sources, each adopting its own alphabet. The main components of the proposed framework are: *(i) a matching schema*, which formalizes matches between symbols, and *(ii)* a *generalized metric function*, which abstracts the computation of string metrics on the basis of a pre-defined matching schema. The proposed framework is based on the identification of the best matching schema for the metric function, i.e., the matching schema leading to the minimum value of the distance function when applied on the strings into consideration. We will prove the hardness of this problem, thus we provide various heuristics in order to approach it, together with a large series of tests. Then, application contexts with various contributions to each of them are presented. We highlight how the proposed framework and the specialization of it in a particular metric can be used in order to approach relevant challenges. Finally, at the end of this thesis, we draw our conclusion and we give space for future works.

The plan of the thesis is as follows: in Chapter 2 we give the background needed and we introduce fundamental concepts such as *strings* and *distances*; moreover, a review of the current related literature is provided. In Chapter 3 we introduce and define the proposed framework: few examples are given, plus we show how the framework generalizes most of the existing string metrics based on symbol identity. In Chapter 4, we formally define the Multi-Parameterized Edit Distance, a generalization of the edit distance with the support of the proposed framework, and we discuss its computational issues. Chapter 5 is devoted to the presentation of the various heuristics designed and implemented in order to approach computational issues of the generalization: first, a survey on heuristics is carried out to acquire a global view of the problem, then three heuristics are selected, ad-hoc implemented and tested out. Chapter 6 is devoted to describe application contexts which have been studied in this thesis. They can be distinguished in two macro areas: engineering, with Wireless Sensors Area Networks, and biomedical informatics, with White Matter Fiber-Bundles analysis and Electroencephalogram analysis. Finally, in Chapter 7 we draw some conclusions and highlight future work.

Chapter 2

Background and Problem Definition

2.1 Introduction

In this chapter we give an overview of the background needed for this thesis. The plan of the chapter is as follows: we first introduce fundamental concepts such as strings and similarity metrics. Then, we exploit the problems that emerge in the general context of heterogeneous alphabets and we point out the problem of discovering hidden correlations. Also, we discuss the context of our problem and its hardness. Finally, we describe an overview of the related academic literature in this context.

2.2 The fundamental concept of string

Strings essentially are ordered sequences of symbols. For a moment, we refrain to give a precise and formal definition for them. Instead, we concentrate on some examples of what a string can be, how can be represented and, most importantly, what does it mean, i.e., what is the semantic of a string. In this thesis, strings are distinguished by using a monospaced font, `such as this one`. Also, terms such as *sequences* and *streams* are synonyms for strings and we will often use them interchangeably in various contexts.

Table 2.1 shows some examples of strings.

Strings are usually drawn from an *alphabet*, which is a finite and nonempty set whose elements are symbols, which are usually also called *characters* or *letters*. Thus, a *string*

String example
`abcdwxyz`
`0422003`
`010001001110`
`--XyX@@%115&&//`
`this phd thesis does not exist`

Table 2.1: Few examples of strings.

defined on an alphabet Σ is a finite sequence of elements of Σ. The appearance of a symbol in a string is called *occurrence*; a symbol could occur one or more times in a string, or even zero. As an example, the alphabet the fifth string in Table 2.1 is defined on could be the set of the lowercase English letters and a space, or even simply the alphabet $\Sigma = \{\texttt{d}, \texttt{e}, \texttt{h}, \texttt{i}, \texttt{n}, \texttt{o}, \texttt{p}, \texttt{s}, \texttt{t}, \texttt{x}, \ \}$ (note the space symbol).

Alphabets can be enhanced and more structural properties can be defined. Let $s =$ `phdthssssshtdhp` an example of string defined on Σ. We denote with Σ^* the set of all strings which can be defined on an alphabet Σ. In this case, $\Sigma^* = \{\epsilon, \texttt{d}, \texttt{dd}, \ldots, \texttt{e}, \ldots\}$, where ϵ is the empty string, i.e., the sequence containing no symbols. Further definitions, which are defined in the next sections, will be used throughout the thesis.

2.3 Similarity, correlation and distance

Given a pair of strings, some interesting questions arise, such as *"are these strings related?"*, *"if they are related, how can we measure this relatedness?"* and *"can we define some sort of measure of (dis)similarity between strings?"*. In particular, such questions give us as results the possibility to extrapolate interesting properties, which could be used in various contexts. As an example, suppose we would need to cluster a set of strings $\mathbb{S}$. A cluster $S \in \mathbb{S}$ in this case would be represented by all of the strings $s \in S$ which are "close" or "similar" to each other. Thus, a definition of similarity is needed.

The good news is that such a defintion, which strictly depends on the approach used to compute it, already exists. The first (and also most famous) approach is that of the *Hamming distance* [55], introduced by Richard Hamming as a geometrical model for error detecting codes. Into the space of $\{0, 1\}^n$ points, Hamming introduced a *distance*, better know by its mathematical term *metric*, $D(x, y)$, where $x, y \in \{0, 1\}^n$. The definition of that distance is based on the observation that a single error in a code point changes one coordinate, two errors, two coordinates, etc. Thus, the distance $D(x, y)$ between two points x and y is defined as the number of coordinates for which x and y are different. $D(x, y)$ is a metric, as shown in [55].

Let $s_1 =$ `0100110` and $s_2 =$ `0110100` be two strings. Clearly, $D(s_1, s_2) = 2$ and we visualize this results by drawing s_1 and s_2 one below the other. Additionally, in the row below the last string we denote with the symbol $\star$ a position in which s_1 and s_2 *match*, e.g., they have a symbol in common.

```
s1 :  0100110
s2 :  0110100
      ** ** *
```

It is straightforward to see that the Hamming distance can be applied only on points defined on the same space, e.g., the number of components of points has to be the same.

What about points defined on different spaces? More general, can Hamming distance be defined on generic strings? It seems the answer is no, thus a different metric is needed to work with.

Back to 1965, Vladimir Iosifovich Levenshtein, working on information theory and error-correcting codes, introduced a metric for codes capable of correcting deletions, insertions and reversal [75]. This was later called *Levenshtein distance* and is informally defined as the minimum number of single-symbol edits, i.e., *edit operations*, which are deletions, insertions and substitutions, required to change one string into the other, where each operation has a cost assigned to. As for an example, let $s_1 =$ `ATGCA` and $s_2 =$ `GGCA` two strings and let $D(x, y)$ the Levenshtein distance between x and y. Suppose that deletions, insertions and substitution respectively have cost equals to 1, 1 and 2. $D(s_1, s_2)$ is 3 and we can (possibly) obtain it by

1. inserting symbol A at beginning of s_2 (cost 1),
2. substituting symbol G (first one) with symbol T in s_2 (cost 2).

It is straightforward to see that numerically $D(s_1, s_2)$ depends on the cost assigned to each edit operation. Moreover, there could be different ways of applying edit operations in order to obtain $D(s_1, s_2)$.

Levenshtein distance belongs to a larger family of distance called *edit distance*. Different definition of an edit distance use different sets of symbol or string operations. Other examples of edit distances are the *longest common subsequence* (LCS) which allows deletions and insertions but substitutions, or the *Jaro distance* [65] which allows only transposition. Also, Hamming distance is an edit distance, allowing only substitutions, thus it can be applied only on strings of the same length. Edit distance essentially represents a parametric metric, equipped with a set of allowed edit operations and to each operation a cost is assigned.

In [113], Wagner and Fischer proposed an algorithm for determining a sequence of edit transformations that changes one string into another, i.e., computing the edit distance, using the same three edit operations used in the Levenshtein distance with cost being parameterized. Their algorithm is proportional to the product of the lengths of the two input strings and it represents one of the earliest problem of dynamic programming studied by all of computer science students.

During this thesis, when it is not differently specified, we will use *(classical) edit distance* and *Levenshtein distance* as synonyms, and we will make use of the algorithm presented in [113] for the implementation. Moreover, we will use the term *alignment* in the classical term, representing a number of edit operations needed to transform a string into another.

2.4 The real problem

As pointed out in the previous sections, various string metrics exist and they may significantly differ for the rules adopted to measure the (dis)similarity degree; it appears obvious that these rules depend on the context in which they are applied. For example, in [43] a survey on duplicate record detection in databases indicates different string metrics with different allowed edit operations which have been specifically tailored for the application context. However, it is important to point out that most of available metrics are based on the natural assumption that *identical symbols among strings represent identical information*, whereas different symbols introduce, in a way or another, some form of differentiation.

Let s = `AAABCD` and t = `111234`: any standard metric would state that they are completely different. Nevertheless, there are cases (like the one shown above) in which symbol identity seems to be not enough. In fact, even if there are no common symbols between two strings, it could happen that they somehow represent similar information.

What happens if we strongly believe that there is some underlying matching between two strings that apparently have different symbols but similar structures? As an example, consider two strings s and t and assume that they are generated by heterogeneous data streams, derived from two different ways of measuring the same reality; think, for instance, of two sensors one measuring light and the other measuring temperature. The values, the scales and the meaning of the two sensors may be very different, but, if sensors are near to a fire, light can be influenced by temperature, and vice versa. To properly monitor this last event, we should, then, be able to understand the correlations between these two heterogeneous measurements. As a further example, suppose that s and t have been deliberately manipulated in such a way as to appear dissimilar, even if their meaning is actually identical - think, for instance, of code cloning techniques [67]. In these cases, the necessity arises of a suitable metric capable of capturing hidden correlations between strings. This metric should take in account that different symbols in the involved strings may express similar concepts.

2.5 Related work

String similarity computation has been a challenging issue in the past literature, and several attempts to face this problem have been presented. However, most of the literature generalizing classical approaches with parameterized alphabets focuses on the pattern matching problem (starting from [10]), where the objective is to seek (exact or approximate) occurrences of a given pattern in a text. This is the context in which parameterized strings were introduced. Here, some of the symbols act as parameters that can be properly substituted at no cost; in approximate pattern matching, patterns have a parameterized match with a text if at most k mismatches occur [77].

A seminal work on this topic is presented in [10]. The approach described in this paper compares parameterized strings. It considers *bijective global* transformation functions allowing *exact p-matches* only. This means that the two strings to match must have the same length; thus, no substitutions or insertions are allowed.

Mismatches are allowed in [57], where the authors face the problem of finding all the locations in a string s for which there exists a *global bijection* π that maps a pattern p into the appropriate substring of s minimizing the Hamming distance. From the function viewpoint, *injective* functions, instead of bijective ones, are considered in [6].

String matching has been extensively used for the clone detection problem, i.e., to check if a code contains two or more cloned parts. In [67], a token-based code clone detection approach is presented. Here, matches between strings are carried out by using a suffix-tree algorithm. The code is tokenized by only one parameter; thus, a match is represented by a one-to-many mapping.

The work in [5] introduces the concept of *generalized function matching* applied to the pattern matching problem in several contexts, like image searching, DNA analysis, poetry and music analysis, etc.

The authors in [44] carry out a comprehensive complexity analysis of the pattern matching problem with parameters (called variables in this paper) with several configurations (e.g., exploiting injective or non-injective functions, allowing or disallowing deletions, etc.). They show that most of the considered variants are NP-complete problems. A detailed survey on parameterized matching appears in [83].

Moving towards string similarity computation, a relevant research issue regards the longest common subsequence problem (hereafter, LCS) and its parameterized versions. Some interesting approaches to facing this problem can be found in [16], [92], and [69]. Specifically, in [16], the definition of arc-annotated sequences is used, [92] considers the gapped version of LCS, whereas in [69] the parameterized version of the LCS problem is considered. Interestingly, LCS allows only insertions and deletions, but no substitutions.

String similarity metrics present important overlap with approximate pattern matching, since one can determine the distance between two strings asking whether there exists an approximate pattern matching with at most k mismatches. However, having a direct approach for measuring the parameterized distance provides obvious benefits.

Few works consider the problem of parameterized distances between strings. In [11], the notion of *p-edit* distance is introduced. It focuses on the edit distance, where allowed edit operations are *insertions*, *deletions* and *exact p-matches*. Mismatches are not allowed. Furthermore, two substrings that participate in two distinct exact p-matches are independent of each other, so that mappings have local validity over substrings not broken by insertions and deletions. In particular, within each of these substrings, the associated mapping function must be bijective. The work presented in [69] extends the approach proposed in [11] by requiring the transformation function to have a global validity; however, it still limits the set of allowed edit operations (in particular, substitutions are not possible).

The work in [50] is based on the approach proposed in [10]; it introduces an order-preserving match, but it limits the number of mismatches to k. In [54], a preliminary approach to a *many-to-many* mapping function for string alignment can be found; it computes alignments between two parameterized strings and gives preferences to alignments on the basis of the co-occurrence frequency.

Moreover, it is worth to recall that edit distance has been exploited in the problem of sequence alignment in the bioinformatics context. Here, two symbolic representations of DNA or protein sequences are aligned in order to identify regions of similarity. The comparison aims at looking for evidence that two sequences have diverged from a common ancestor by a process of mutation and selection [39]. Classical edit operations here are intended as the basic mutational processes that are insertions and deletions (also called gaps), which add or remove residues, and substitutions, which change residues in a sequence. Alignments in pairwise alignment are scored by the sum of terms for each aligned pair of residues and each aligned residue pair is scored by using a scoring matrix. Scoring matrices, also called substitution matrices, are matrices which store the score assigned to a pair of aligned residues. Two main categories of scoring matrices exist in literature, namely *(i)* position-independent and *(ii)* position-specific scoring matrices. The former category includes well-known scoring matrices such as PAM and BLOSUM [60, 100], while the latter includes scoring matrices used in protein BLAST [4]. Scoring matrices in both categories are defined upon specific alphabets, e.g., nucleotides or amino acids, and over homogeneous sequences, thus the heterogeneous case is not applicable.

Chapter 3

Generalizing identity-based string comparison metrics

3.1 Introduction

In this chapter, we formally introduce the theoretical foundation of our work. In particular, we define and analyze the framework $\mathfrak{F}$, which is intended to generalizing identity-based string similarity metrics. We first give the formal basis by defining important preliminary concepts and then we introduce *matching schemas* and *generalized metric functions*, which are the main ingredients for the framework $\mathfrak{F}$. Finally, we show how the introduced framework can be exploited to generalize some notable string similarity metrics. Each section is correlated by various examples.

Part of the work proposed in this chapter has been published in [30].

3.2 Preliminaries

In a general context, let Π_1 and Π_2 be two (possibly disjoint) alphabets of symbols and let s_1 and s_2 be two strings defined over Π_1 and Π_2, respectively. We denoted the *length* of a string s, i.e., the number of its symbols, by $len(s_1)$. We refer to a symbol in s by accessing its position: for each position $1 \leq j \leq len(s)$, the j-th symbol of s will be identified by $s_i[j]$ and we will denote the substring of s starting at position x and ending at position y as $s[x..y]$.

In order to define the basis for the framework $\mathfrak{F}$, we need further concepts. The idea is to use a special concept called *matching schema* which intuitively represents how different combinations of the alphabets Π_1 and Π_2 can be combined via matching. A matching schema is extremely important for the definition of $\mathfrak{F}$.

Definition 1 (π-partition) Given an alphabet Π and an integer π such that $0 < \pi \leq |\Pi|$, a *π-partition* is a partition Φ^π of Π such that $0 < |\phi_v| \leq \pi$, for each $\phi_v \in \Phi^\pi$. $\square$

Definition 2 ($\langle\pi_1, \pi_2\rangle$-matching schema) Given two alphabets Π_1 and Π_2 and two integers π_1 and π_2, *a $\langle\pi_1, \pi_2\rangle$-matching schema* is a function $M_{\langle\pi_1,\pi_2\rangle} : \Phi_1^{\pi_1} \times \Phi_2^{\pi_2} \rightarrow \{\mathit{true}, \mathit{false}\}$, where $\Phi_i^{\pi_i}$ ($i \in \{1, 2\}$) is a π_i-partition of Π_i and, for each $\phi_v \in \Phi_1^{\pi_1}$ (resp., $\phi_w \in \Phi_2^{\pi_2}$), there is at most one $\phi_w \in \Phi_2^{\pi_2}$ (resp., $\phi_v \in \Phi_1^{\pi_1}$) such that $M(\phi_v, \phi_w) = \mathit{true}$. This means that all the symbols in ϕ_v match with all the ones in ϕ_w. $M(\phi_v, \phi_w) = \mathit{false}$ indicates that all the symbols in ϕ_v mismatch with all the ones in ϕ_w. $\square$

Intuitively, a π-partition Φ^π is a subset of $\mathcal{P}(\Pi)$ where each subset $\phi_v \in \Phi_\pi$ contains at most π symbols from Π and for each $\phi_v, \phi_w \in \Phi_\pi, \phi_v \cap \phi_w = \emptyset$, i.e., they contain no common symbols. Using two π-partitions, given two strings s_1 and s_2 defined over two alphabets Π_1 and Π_2, a *$\langle\pi_1, \pi_2\rangle$-matching schema* states which symbols of s_1 can be considered matching with symbols of s_2. It is essential to note that many-to-many matching are actually expressed with π-partitions and at the same time they disallow ambiguous matchings.

Example 1 Let $\Pi_1 = \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}\}$ and $\Pi_2 = \{\mathtt{E}, \mathtt{F}, \mathtt{G}, \mathtt{H}\}$. Let $s_1 = \mathtt{AAABCCDCAA}$ and $s_2 = \mathtt{EEFGHGGFHH}$. The values of π_1 and π_2 define the cardinality of each subset in a π-partition. For $\pi_1 = \pi_2 = 2$, one (of the many) possible matching schemas is {{A,B}-{E,H}, {C,D}-{G,F}}. Note that here {A,B}-{E,H} means that symbols A and B match with symbols E and H. As a further example, having $\pi_1 = \pi_2 = 1$, a possibile matching schema is {{A}-{E}, {B}-{G}, {C}-{F}, {D}-{H}}.

It is interesting to point out that, given Π_1, Π_2, π_1 and π_2, many possible matching schemas can be defined. However, in some contexts, it may be useful to limit valid matching schemas via some constraints (a real example is given in Section 6.3). In order to provide this possibility, the following definition formally introduces both the notions of constraint(s) over a matching schema and constrained matching schema.

Definition 3 ($\langle\pi_1,\pi_2,\chi\rangle$-constrained matching schema) *A constraint* χ associated with a matching schema $M_{\langle\pi_1,\pi_2\rangle}$ is a set of unordered pairs of symbols (c_i, c_j), such that $c_i \in \Pi_1$, $c_j \in \Pi_2$ and, for each $(c_i, c_j) \in \chi$, there exists no pair (ϕ_v, ϕ_w), $\phi_v \in {\Phi_1}^{\pi_1}$, $\phi_w \in {\Phi_2}^{\pi_2}$, having $c_i \in \phi_1, c_j \in \phi_2$ and $M(\phi_1, \phi_2) = \mathit{true}$. A $\langle\pi_1,\pi_2,\chi\rangle$*-constrained matching schema* is represented by $M_{\langle\pi_1,\pi_2,\chi\rangle}$ only if $\chi \neq \emptyset$. □

Example 2 Continuing the Example 1, suppose we have $\chi = \{(\mathtt{B},\mathtt{H}),(\mathtt{C},\mathtt{F})\}$, the matching schema represented by {{A,B}-{E,H}, {C,D}-{G,F}} is no more valid, whereas the matching schema represented by {{A,B}-{E,F}, {C,D}-{G,H}} is.

Throughout the whole thesis, whenever it is clear from the context, for the sake of simplicity, we avoid to write $M_{\langle\pi_1,\pi_2,\chi\rangle}$ and we simply denote it by M.

The generalizability of the framework $\mathfrak{F}$ is based on the fact that it can generalize any string metric based on the assumption of symbol identity. Therefore, in defining it, we are not interested in a particular function specification, rather in a generalization of a metric function.

Definition 4 (Generalized metric function) Given a metric function $f(\cdot,\cdot)$, based on symbol identity, and given a valid constrained matching schema M, the *generalized metric function* $f^{M_{\langle\pi_1,\pi_2,\chi\rangle}}(\cdot,\cdot)$ (or simply $f^M(\cdot,\cdot)$) is obtained from $f(\cdot,\cdot)$ by substituting symbol identity with the symbol matchings defined in M. □

Having a generalized metric function, we now need to formally define the result of the application of it, i.e., the distance given by the application of the metric. In this case, the obtained distance depends on M and f.

Definition 5 (Generalized distance) Given two strings s_1 and s_2 over Π_1 and Π_2, respectively, and given the set $\mathbb{M}$ of valid constrained matching schemas, the *generalized distance* $F(s_1, s_2)$ between s_1 and s_2 returns the minimum value returned by $f^M(s_1, s_2)$ that can be obtained by taking any possible matching schema M of $\mathbb{M}$. Formally:

$$F(s_1, s_2) = \min_{M_{\langle\pi_1,\pi_2,\chi\rangle} \in \mathbb{M}} \{f^M(s_1, s_2)\}.$$

□

Finally, we defined all of the bricks needed in order to build the framework $\mathfrak{F}$. In Section 3.3 we formally define the framework and we give the intuition behind it.

3.3 The framework

Thanks to the definitions in Section 3.2, we are now able to present our framework $\mathfrak{F}$. It consists of a quintuple:

$$\mathfrak{F} = \langle \Pi_1, \Pi_2, \langle \pi_1, \pi_2, \chi \rangle, \mathbb{M}, f^M(\cdot, \cdot) \rangle$$

where Π_1 and Π_2 are the alphabets on which the strings under consideration are defined, $\langle \pi_1, \pi_2, \chi \rangle$ are the parameters necessary to define valid matching schemas, $\mathbb{M}$ is the set of all the valid constrained matching schemas over Π_1 and Π_2, and $f^M(\cdot, \cdot)$ is the generalized metric function. When applied on two strings s_1 and s_2 over the alphabets Π_1 and Π_2, respectively, $\mathfrak{F}$ returns the value of $F(s_1, s_2)$, where $F(s_1, s_2)$ is the generalized distance of s_1 and s_2 over $f(\cdot, \cdot)$ and $\mathbb{M}$.

3.4 Generalization of notable string similarity metrics

In the previous sections, we introduced our framework $\mathfrak{F}$, which paves the way to a quite general computation of string (dis)similarity. We are now ready to show that $\mathfrak{F}$ is general enough to encompass several classical and notable string similarity metrics. Our idea is to first show that a particular specialization of $\mathfrak{F}$ includes both character-based and token-based distances; starting from this, we show that $\mathfrak{F}$ can also be specialized to more sophisticated comparison approaches, such as *parameterized pattern matching* proposed in [10].

Proposition 1 Given the following specialization of $\mathfrak{F}$:

1. $\pi_1 = \pi_2 = 1$;

2. $\chi = \{(c_i, c_j) | c_i \in \Pi_1, c_j \in \Pi_2, c_i \neq c_j\}$;

then, $f^M(\cdot, \cdot) = f(\cdot, \cdot)$ for all the metric functions $f(\cdot, \cdot)$ based on symbol identity. $\square$

The reason behind the Proposition 1 is that, intuitively, with $\pi_1 = \pi_2 = 1$, the cardinality of each matching subset is equal to 1, i.e., we admit only one-to-one symbol matchings.

Furthermore, with the provided construction of χ, only symbol identities are allowed. As a consequence, there is only one valid matching schema $M_{\langle 1,1,\chi \rangle}$, which is the one stating that, for each pair $(\phi_v, \phi_w), \phi_v \in \Phi_1^{\pi_1}$ and $\phi_w \in \Phi_2^{\pi_2}$, $M(\phi_v, \phi_w) = \mathit{true}$ if and only if $\phi_v = \phi_w$. This matching schemas simulates symbol identity specification for the generalized metric functions based on symbol identities.

In the following subsections, we show how our framework $\mathfrak{F}$, according to Proposition 1, can be used to generalize some notable string similarity metrics.

3.4.1 Edit Distance

In its simplest form, the edit distance between two strings s_1 and s_2 is the minimum number of edit operations (insertions, deletions or substitutions) of single characters needed to transform s_1 into s_2, where each operation has a cost equal to 1 [75]. This version of the edit distance is also referred to as Levenshtein distance. A basic algorithm for edit distance computation exploits a dynamic programming approach; in it, the choice of the edit operation is carried by a recurrence formula, where the discriminating factor is based on the question "Is $s_1[i] = s_2[j]$?".

If our framework $\mathfrak{F}$ is applied, this question is substituted by the question "According to M, symbols $s_1[i]$ and $s_2[j]$ should be matched?". As specified by Proposition 1, according to the specialization defined therein, this question is equivalent to asking for symbol identity.

3.4.2 Affine Gap Distance

The affine gap distance metric [116] is similar to the edit distance, except for the fact that it introduces two extra edit operations, i.e., *open gap* and *extend gap*. In this way, the cost of the first insertion (gap opening) can be different from, and is usually higher than, the cost of adding consecutive insertions (gap extensions). Similarly to the edit distance, the discriminating factor in computing the affine gap distance is based on the question "Is $s_1[i] = s_2[j]$?". When applying our framework $\mathfrak{F}$ with the specialization introduced in Proposition 1, this question is substituted by the one "According to M, symbols $s_1[i]$ and $s_2[j]$ should be matched?", which has been shown to be equivalent to asking for symbol identity.

3.4.3 Smith-Waterman Distance

The Smith-Waterman distance [104] is an extension of both the edit distance and the affine gap distance, in which mismatches at the beginning and the end of strings have lower costs than mismatches in the middle. This metrics allows for substring matchings and is well suited for fitting shorter strings into longer ones. Also in this case, the basic resolution schema exploits dynamic programming, where the discriminating factor in the recurrence formula is the question "Is $s_1[i] = s_2[j]$?".

3.4.4 Jaro Distance Metric

The Jaro distance metric, introduced in [65], is based on the concepts of common characters and transpositions.

Given two strings s_1 and s_2, two symbols $s_1[i]$ and $s_2[j]$ are *common characters* when $s_1[i] = s_2[j]$ and $|i - j| \le \frac{1}{2} \min\{len(s_1), len(s_2)\}$. Given the i-th common character a in s_1 and the j-th common character b in s_2, if $a \neq b$ this is a transposition.

The Jaro distance value is, then, computed as:

$$\text{Jaro}(s_1, s_2) = \frac{1}{3}\left(\frac{c}{len(s_1)} + \frac{c}{len(s_2)} + \frac{c - \frac{1}{2}t}{c}\right)$$

where c is the number of common characters and t is the number of transpositions.

The characteristics of the Jaro distance metric clearly differ from the ones of the aforementioned metrics. However, as a common core for the definitions of common character and transposition, there is, again, symbol equality, where the discriminating question is "Is $s_1[i] = s_2[j]$?".

Also in this case, it is easy to show that substituting this question with the one "According to M, symbols $s_1[i]$ and $s_2[j]$ should be matched?", under the specialization of Proposition 1, makes the two settings equivalent.

3.4.5 Atomic Strings

In *atomic strings* [86], the comparison shifts away from single character to longer strings. In particular, s_1 and s_2 are tokenized by punctuation characters. Each token is called *atomic string*. Two atomic strings match if they are equal or if one is a prefix of the other. The similarity between s_1 and s_2 is, then, computed as the fraction of the atomic strings that match.

Observe that, even if this metrics moves from the comparison of single characters to the one of substrings, the basic operation used to identify a matching for atomic strings is the identity of sequences of single characters. As a consequence, the same considerations outlined above when substituting the question “Is $s_1[i] = s_2[j]$?” with the one “According to M, symbols $s_1[i]$ and $s_2[j]$ should be matched?” are still valid.

3.4.6 WHIRL

In [33] the cosine similarity is combined with the *tf.idf* weighting scheme in the WHIRL system to compare pairs of strings in a set of records. In particular, each string s is separated into words; a weight $v_s(w)$ is assigned to each word w of s, which depends on the number tf_w of times when w appears in s and on the fraction idf_w of records containing w. The cosine similarity between two strings s_1 and s_2 is, then, defined as:

$$\mathrm{sim}(s_1, s_2) = \frac{\Sigma_w v_{s_1}(w) \cdot v_{s_2}(w)}{||v_{s_1}||^2 \cdot ||v_{s_2}||^2}.$$

Despite the complexity of this metric, as for its application to our context, the most relevant thing to observe is that both the tf_w and the idf_w components are based on the exact occurrence of w in the string(s). As a consequence, again, when asking whether a word w is contained into a string s, the basic question for the computation is “Is $w[i] = s[j]$?”, which, as previously shown, can be simulated by the question “According to M, symbols $w[i]$ and $s[j]$ should be matched?”, under the specialization of Proposition 1.

3.4.7 Q-grams with *tf.idf*

Q-grams with *tf.idf* [53] extends the metric adopted in WHIRL by using q-grams, instead of words. This allows the management of spelling errors, the insertion and the deletion of words. The computation setting is equal to the one of WHIRL; therefore, all considerations about the specialization of $\mathfrak{F}$ seen for WHIRL can be applied also to this metrics.

3.4.8 Parameterized pattern matching

In [10], an approach to comparing strings over partially overlapping alphabets is proposed. This approach, named *parameterized pattern matching*, aims at identifying pairs of strings being equal except for a one-to-one symbol substitution. In particular, the alphabet of each string s_i is partitioned in two alphabets, namely Σ_i and Π_i, the former containing standard symbols and the latter encompassing parameters. Parameters can be renamed at no cost. Two such strings identify a parameterized match if one string can be obtained by renaming the parameters of the other by means of a one-to-one function. This approach has been shown to be particularly useful in code cloning identification.

Our framework can be specialized to accommodate parameterized pattern matching. In fact, given two strings s_1 and s_2, defined over the alphabets $\Sigma_1 \cup \Pi_1$ and $\Sigma_2 \cup \Pi_2$, respectively, consider the following specialization of $\mathfrak{F}$:

1. $\pi_1 = \pi_2 = 1$;
2. $\chi = \{(c_i, c_j) | c_i \in \Sigma_1, c_j \in \Sigma_2, c_i \neq c_j\}$;

Let $f(\cdot,\cdot)$ be the Hamming distance (which, basically, is the edit distance allowing symbol substitutions only). If $F(s_1, s_2) = 0$, then there is a parameterized matching between s_1 and s_2.

In particular, $\pi_1 = \pi_2 = 1$ constraints to one-to-one functions. The definition of χ states that the only valid match configuration between pairs of symbols in Σ_1 and Σ_2 is symbol identity, whereas any symbol in Π_1 (resp., in Π_2) can be matched at no cost, by means of a one-to-one matching function, with any symbol of the other string. $F(\cdot,\cdot)$ finds the minimum value of the Hamming distance, among all the possible one-to-one substitutions. A value of this distance equal to 0 denotes that one string can be obtained by renaming the parameters of the other by means of a one-to-one function and, consequently, that a parameterized match holds.

Chapter 4

Multi-Parameterized Edit Distance

4.1 Introduction

In this chapter we formally define one of the most important result derived from the framework $\mathfrak{F}$, showing how it can be applied to generalize a classic metric in such a way that symbol identity is substituted by many-to-many symbol correlations, with respect to the general idea and motivation behind it given in Chapter 1, where identifying the best matching schema is part of the problem. Furthermore, this chapter focuses on studying in detail all of the theoretical and practical implications of this generalization. We introduce the *Multi-Parameterized Edit Distance* (MPED), which is a generalization of the classical edit distance with the support of our framework. The MPED allows the computation of the minimum edit distance between two strings, provided that finding the optimal matching schema, under a set of constraints, is part of the problem. This chapter is organized as follows: in Section 4.2 we introduce some basic definitions and the definition of MPED. Then, in Section 4.3 some examples are presented in order to give to the reader some insights on the problem in instance. Finally, in Section 4.4 computational issues are discussed. In particular, the NP-Hardness of the problem and a lower bound L are analyzed in depth.

Part of the work presented in this chapter is included in [30].

4.2 Definitions

Definition 6 (Transposition) Let s_1 and s_2 be two strings defined over the alphabets Π_1 and Π_2. Let $-$ be a symbol not included in $\Pi_1 \cup \Pi_2$. Then, a string $\bar{s}_i$ over $\Pi_i \cup \{-\}$

($i \in 1, 2$) is a *transposition* of s_i if $\bar{s}_i$ can be obtained from s_i by deleting all the occurrences of $-$. The set of all the possible transpositions of s_i is denoted by $\mathcal{TR}(s_i)$. □

Definition 7 (Alignment) An *alignment* for the strings s_1 and s_2 is a pair $\langle \bar{s}_1, \bar{s}_2 \rangle$, where $\bar{s}_1 \in \mathcal{TR}(s_1)$, $\bar{s}_2 \in \mathcal{TR}(s_2)$ and $len(\bar{s}_1) = len(\bar{s}_2)$. Here, $-$ is meant to denote an insertion/deletion operation performed on s_1 or s_2. □

Definition 8 (Match and distance) Let $\langle \bar{s}_1, \bar{s}_2 \rangle$ be an alignment for s_1 and s_2, let $M_{\langle \pi_1, \pi_2, \chi \rangle}$ be a $\langle \pi_1, \pi_2, \chi \rangle$-constrained matching schema over π-partitions $\Phi_1^{\pi_1}$ and $\Phi_2^{\pi_2}$ and the set of constraints χ, and let j be a position with $1 \leq j \leq len(\bar{s}_1) = len(\bar{s}_2)$. We say that $\langle \bar{s}_1, \bar{s}_2 \rangle$ has a match at j if:

- $s_1[j] \in \phi_v, s_2[j] \in \phi_w$, $\phi_v \in \Phi_1^{\pi_1}$, $\phi_w \in \Phi_2^{\pi_2}$ and $M_{\langle \pi_1, \pi_2, \chi \rangle}(\phi_v, \phi_w) = true$.

The *distance* between $\bar{s}_1$ and $\bar{s}_2$ under $M_{\langle \pi_1, \pi_2, \chi \rangle}$ is the number of positions at which the pair $\langle \bar{s}_1, \bar{s}_2 \rangle$ does not have a match. □

Given the previous definitions, we can introduce the notion of *Multi-Parameterized Edit Distance* between two strings s_1 and s_2 as follows:

Definition 9 (Multi-Parameterized Edit Distance - MPED) Let π_1 and π_2 be two integers such that $0 < \pi_1 \leq |\Pi_2|$ and $0 < \pi_2 \leq |\Pi_1|$; the Multi-Parameterized Edit Distance between s_1 and s_2 ($\mathcal{L}_{\langle \pi_1, \pi_2, \chi \rangle}(s_1, s_2)$, for short) is the minimum distance that can be obtained with any $\langle \pi_1, \pi_2, \chi \rangle$-constrained matching schema and any alignment $\langle \bar{s}_1, \bar{s}_2 \rangle$.

Formally:

$$\mathfrak{F}_{\mathcal{L}} = \langle \Pi_1, \Pi_2, \langle \pi_1, \pi_2, \chi \rangle, \mathbb{M}, \mathcal{L}^M(\cdot, \cdot) \rangle$$

and

$$F_{\mathcal{L}}(s_1, s_2) = \mathcal{L}_{\langle \pi_1, \pi_2, \chi \rangle}(s_1, s_2) = \min_{M_{\langle \pi_1, \pi_2, \chi \rangle} \in \mathbb{M}} \{\mathcal{L}^M(s_1, s_2)\}.$$

where $\mathcal{L}(\cdot, \cdot)$ is the classical edit distance. □

Observe that, in order to properly compute $\mathcal{L}_{\langle \pi_1, \pi_2, \chi \rangle}(s_1, s_2)$, several components play a crucial role, namely:

- π_1 and π_2, which determine the (maximum) size of each partition;
- π-partitions $\Phi_1^{\pi_1}$ and $\Phi_2^{\pi_2}$; in fact, there can be many π-partitions for the same set of π_1, π_2, Π_1, and Π_2;
- matching schemas $M_{\langle \pi_1,\pi_2,\chi\rangle}$, which determine the way to combine partitions of different sets via matching;
- alignments; in fact, there can be many possible alignments between two strings.

Moreover, note how χ plays a crucial role in the computation of MPED. In fact, when χ is not empty, it means that additional information about which symbols should not match is injected in the computation. In order to reinforce this concept and to point out that some *a priori* knowledge might be available, even if not complete, we enrich the notion of MPED with a particularly common case, i.e., when identical symbols should be considered as matching anyway, yet allowing many-to-many matching in the matching schema. We call this specific case *Semi Blind Multi-Parameterized Edit Distance* and we formally define it enriching Definition 8 as follows:

Definition 10 (Semi-Blind Multi-Parameterized Edit Distance - MPED$_{SB}$) The Semi-Blind Multi-Parameterized Edit Distance (MPED$_{SB}$ for short) is obtained from MPED by considering a match between $\langle \overline{s_1}, \overline{s_2}\rangle$ at j if (cfr Definition 8):

- $s_1[j] \in \phi_v, s_2[j] \in \phi_w$, $\phi_v \in \Phi_1^{\pi_1}$, $\phi_w \in \Phi_2^{\pi_2}$ and $M_{\langle \pi_1,\pi_2,\chi\rangle}(\phi_v, \phi_w) = true$,
- $s_1[j] = s_2[j]$.

Observe that this concept might be further generalized with matchings that should always hold (as opposit to constraints, specifying matchings that must not hold). However, this is a less interesting case, in practice.

4.3 Examples

Example 3 Let $s1$ = `AAABCCDCAA` and $s2$ = `EEFGHGGFHH`, which determines $\Pi_1 = \{$`A`,`B`,`C`,`D`$\}$ and $\Pi_2 = \{$`E`,`F`,`G`,`H`$\}$. For $\pi_1 = \pi_2 = 1$, the best alignment $\langle \bar{s}_1, \bar{s}_2\rangle$ that can be computed is obtained by matching $\{$`A`$\}$–$\{$`E`$\}$, $\{$`B`$\}$–$\{$`G`$\}$, $\{$`C`$\}$–$\{$`H`$\}$, and $\{$`D`$\}$–$\{$`F`$\}$. The aligment:

```
s_1 : AAABCCDDCAA → AAABCCDDCAA
s_2 : EEFGHGGFHH  → EEFGHGGFH-H
                    ** **   **
```

gives $\mathcal{L}_{\langle 1,1\rangle}(s_1, s_2) = 5$. Observe that this approach works properly even if $\Pi_1 \cap \Pi_2 = \emptyset$ and the input strings have different lengths.

If we set $\pi_1 = \pi_2 = 2$, the best alignment is the one obtained by matching {B, A}-{E, H}, and {C, D}-{G, F}, namely:

```
s_1 : AAABCCDDCAA → AAABCCDDCAA
s_2 : EEFGHGGFHH  → -EEFGHGGFHH
                     ** * *****
```

which gives $\mathcal{L}_{\langle 2,2\rangle}(s_1, s_2) = 3$.

Suppose, now, to introduce the constraint $\chi = \{\langle \mathtt{A}, \mathtt{E}\rangle\}$. For $\pi_1 = \pi_2 = 1$, the best alignment $\langle \bar{s}_1, \bar{s}_2\rangle$ is the following:

```
s_1 : AAABCCDDCAA → AAABC-CDDCAA
s_2 : EEFGHGGFHH  → --EEFGHGGFHH
                       **  *****
```

which still gives $\mathcal{L}_{\langle 1,1,\chi\rangle}(s_1, s_2) = 5$, but where A and E do not match anymore. Optimal matchings are, in fact, {A}-{H}, {B}-{E}, {C}-{F}, and {D}-{G}. □

4.4 Computational issues: complexity

4.4.1 NP-Hardness

After having defined MPED, it is important to analyze its properties. We start by determining its computational complexity. The specialization of $\mathfrak{F}$ provided in Proposition 1 ensures that, in that case, there exists only one valid matching schema in $\mathbb{M}$, and this can be easily derived by construction. However, in general, $\mathbb{M}$ could contain n matching schemas; each matching schema $M_{\langle \pi_1, \pi_2, \chi\rangle} \in \mathbb{M}$ could match π_1 symbols of Π_1 with π_2 symbols of Π_2; furthermore, for each matching schema, several alignments between s_1 and s_2 are possible.

In this section, we show that the general problem of computing MPED is NP-Hard.

Theorem 1 The problem of computing $\mathcal{L}_{\langle \pi_1, \pi_2, \chi \rangle}(s_1, s_2)$ is NP-Hard.

Proof.

This theorem can be proven by performing a reduction from the three-dimensional matching problem (hereafter, 3DM) [48]. First recall that 3DM is defined as follows:

> Let X, Y and Z be finite and disjoint sets, and let M be a subset of $X \times Y \times Z$. That is, M consists of triples $\langle X_i, Y_i, Z_i \rangle$ such that $X_i \in X$, $Y_i \in Y$, and $Z_i \in Z$. Now, $M' \subseteq M$ is a 3DM if, for any two distinct triples $\langle X_1, Y_1, Z_1 \rangle \in M'$ and $\langle X_2, Y_2, Z_2 \rangle \in M'$, we have $X_1 \neq X_2$, $Y_1 \neq Y_2$, and $Z_1 \neq Z_2$.

The decision problem is, then, stated as follows: given a set M and an integer k, decide whether there exists a 3DM $M' \subseteq M$ such that $|M'| \geq k$.

Let us, now, turn to the reduction. Let M_i indicate the i-th triple in M and let $q = |M|$. Let $\Pi_1 = X \cup Y \cup Z \cup \Sigma$ and $\Pi_2 = \{M_i\} \cup \Sigma$, where the pairs (X_i, M_j) (resp., (Y_i, M_j), (Z_i, M_j)) are not constrained, i.e., any of these pairs can possibly match.

For each triple M_i, Σ contains the set of symbols $\{c_{i_1} \ldots c_{i_q},\ t_{i_1} \ldots t_{i_7},\ m_{i_1} \ldots m_{i_7}\}$; the set of constraints χ is constructed in such a way as to allow symbol identity only among the symbols in $\{c_{i_1} \ldots c_{i_q},\ t_{i_1} \ldots t_{i_7}, m_{i_1} \ldots m_{i_7}\}$.

Then, starting from M, we build the following strings:

$$s_1 = \Phi_1 \quad c_{1_1}\ c_{1_2}\ \cdots\ c_{1_q} \quad \Phi_2 \quad c_{2_1}\ c_{2_2}\ \cdots\ c_{2_q}\ \cdots\ \Phi_q \quad c_{q_1}\ c_{q_2}\ \cdots\ c_{q_q}$$

$$s_2 = \Psi_1 \quad c_{1_1}\ c_{1_2}\ \cdots\ c_{1_q} \quad \Psi_2 \quad c_{2_1}\ c_{2_2}\ \cdots\ c_{2_q}\ \cdots\ \Psi_q \quad c_{q_1}\ c_{q_2}\ \cdots\ c_{q_q}$$

where, for each triple M_i, the "blocks" Φ_i and Ψ_i have the form:

$$t_{i_1} \quad X_i \quad t_{i_2} \quad t_{i_3}\ t_{i_4} \quad Y_i \quad t_{i_5} \quad t_{i_6}\ t_{i_7} \quad Z_i$$

$$M_i\ m_{i_1}\ m_{i_2}\ M_i\ m_{i_3}\ m_{i_4}\ m_{i_5}\ M_i\ m_{i_6}\ m_{i_7}$$

To clarify this construction, consider the set $M = \{\langle X_1, Y_2, Z_2 \rangle, \langle X_2, Y_1, Z_1 \rangle, \langle X_3, Y_2, Z_3 \rangle, \langle X_3, Y_3, Z_4 \rangle\}$. The first triple generates the following portions of strings:

$$s_1 = t_{1_1} \quad X_1 \quad t_{1_2} \quad t_{1_3} \; t_{1_4} \quad Y_2 \quad t_{1_5} \quad t_{1_6} \; t_{1_7} \quad Z_2 \quad c_{1_1} \; c_{1_2} \; c_{1_3} \; c_{1_4} \; \cdots$$

$$s_2 = M_1 \; m_{1_1} \; m_{1_2} \; M_1 \; m_{1_3} \; m_{1_4} \; m_{1_5} \; M_1 \; m_{1_6} \; m_{1_7} \; c_{1_1} \; c_{1_2} \; c_{1_3} \; c_{1_4} \; \cdots$$

Observe that, without any edit operation on s_1 and s_2, the $c_{i_1} \cdots c_{i_q}$ blocks match, whereas all the other blocks (each consisting of 10 symbols) do not match. As a consequence, without any edit operation, the distance between the two strings is $d = 10 \cdot q$. For instance, in the previous example, the initial distance is $d = 40$.

Assume, now, that we are interested in computing $\mathcal{L}_{\langle \pi_1, \pi_2, \chi \rangle}(s_1, s_2)$, where $\pi_1 = 1$ and $\pi_2 = 3$. In other words, assume that each parameter in s_1 can match at most one parameter in s_2[1], whereas each parameter in s_2 can match up to 3 parameters in s_1[2].

First we focus on the alignment of the blocks Φ_i and Ψ_i; it is easy to see that, if these blocks are considered in isolation, the only way to reduce their edit distance is to assign both X_i, Y_i, and Z_i to M_i in the matching schema and align them as follows:

$$\begin{array}{cccccccccccccc} t_{i_1} & X_i & t_{i_2} & t_{i_3} & t_{i_4} & Y_i & t_{i_5} & t_{i_6} & t_{i_7} & Z_i & - & - \\ - & M_i & m_{i_1} & m_{i_2} & - & M_i & m_{i_3} & m_{i_4} & m_{i_5} & M_i & m_{i_6} & m_{i_7} \end{array}$$

In this case, the distance between these two blocks becomes 9.

It is easy to check that, if at least one among X_i, Y_i, and Z_i is not associated with M_i in the matching schema, or a different alignment is carried out among blocks, the obtained distance is greater than or equal to 10. As a consequence, the corresponding choice is not convenient on the global perspective of finding the minimum edit distance.

Now, observe that the role of the blocks $c_{i_1} \cdots c_{i_q}$ is exactly to isolate the Φ_i-Ψ_i blocks; in fact, it is easy to verify that it will be never convenient to perform insertions and deletions within these blocks, since these operations would increase the overall distance.

In conclusion, $\mathcal{L}_{\langle \pi_1, \pi_2, \chi \rangle}(s_1, s_2)$ is obtained in correspondence of the matching schema that selects the maximum number of disjoint triples. The number of these triples can be determined by the decrease of the initial distance (each selected triple implies that distance decreases by 1). As a consequence, given a set M, such that $|M| = q$, and an integer k, deciding whether there exists a 3DM $M' \subseteq M$ such that $|M'| \geq k$ corresponds to checking

[1]This corresponds to say that each X_i (resp., Y_i, Z_i) can be selected in at most one triple.
[2]This is needed to accommodate the triple $\langle X_i, Y_i, Z_i \rangle$ in M_i.

whether $\mathcal{L}_{\langle\pi_1,\pi_2,\chi\rangle}(s_1, s_2) \leq 10 \cdot q - k$, with $\pi_1 = 1$ and $\pi_2 = 3$. This implies that 3DM can be reduced to the computation of $\mathcal{L}_{\langle\pi_1,\pi_2,\chi\rangle}(s_1, s_2)$ and, therefore, that this last task is NP-Hard. $\square$

4.4.2 A lower bound L

The intractability of the problem of computing $\mathcal{L}_{\langle\pi_1,\pi_2,\chi\rangle}(s_1, s_2)$ points out the need of heuristic approaches for its solution. In this context, it is highly useful to establish a lower bound L for $d^* = \mathcal{L}_{\langle\pi_1,\pi_2,\chi\rangle}(s_1, s_2)$ that could be computed in polynomial time; indeed, L could be used to evaluate the quality of the results returned by heuristic approaches.

First, we start with the definition of a lower bound L for d^* when $\pi_1 = 1$ and $\pi_2 = 1$, and, then, we generalize it to any value of π_1 and π_2. We point out that the computation of L is not a trivial task even for the case when $\pi_1 = 1$ and $\pi_2 = 1$; in fact, a trivial lower bound could be easily set to 0, but a useful value for L should approximate d^* as much as possible.

Given two strings s_1 and s_2 over Π_1 and Π_2, respectively, our goal is to characterize d^* as precisely as possible by estimating the role, in the computation of d^*, of each pair of symbols (a, b) such that $a \in \Pi_1$ and $b \in \Pi_2$, without the need of computing the optimal matching schema first. Unfortunately, the role played by each pair (a, b) is not independent of the matchings of the other pairs of symbols included in the optimal matching schema. In particular, it is not possible to compute the exact role played by (a, b) without preliminarily fixing the whole matching schema.

As a consequence, our aim is to estimate d^* on the basis of an estimation of the maximum number of potential matchings between pairs of symbols, given a maximum number of insertions and deletions (*indels*, in the following) performed on s_1 or s_2.

Without loss of generality, to simplify the presentation, in the following, we assume that $|\Pi_1| = |\Pi_2| = p$ and $len(s_1) = len(s_2) = l$.

To describe our approach for the computation of L, we observe that d^* could be written as:

$$d^* = l + \delta^* - \mu^*$$

where δ^* is the number of indels present in the optimal alignment $\langle \bar{s}_1, \bar{s}_2 \rangle$, and μ^* is the number of matches present in $\langle \bar{s}_1, \bar{s}_2 \rangle$. In the following, we will denote as $\langle \bar{s}_1, \bar{s}_2 \rangle_\delta$ the optimal alignment that can be obtained allowing at most δ indels.

Clearly, δ^* and μ^* are not known a priori. However, we can polynomially simulate the computation of d^* as follows: for increasing values δ of indels we estimate the number μ_δ of matches that can be obtained. The minimum value of the formula $l + \delta - \mu_\delta$ is, then, the best estimation of d^*. Formally:

$$L = \min_{\delta \in [0..l/2]} \{l + \delta - \mu_\delta\}$$

Now, given a maximum number δ of indels allowed in the alignment, we compute μ_δ by estimating the number of potential matches $\omega_\delta(a, b)$ for each pair of symbols (a, b) in the corresponding alignment $\langle \bar{s}_1, \bar{s}_2 \rangle_\delta$ as follows:

$$\omega_\delta(a, b) = \min\left(\sum_{i=1}^{l} \kappa(a, b, s_1[i], s_2[i - \delta..i + \delta]), \sum_{i=1}^{l} \kappa(a, b, s_1[i - \delta..i + \delta], s_2[i])\right)$$

where $\kappa(a, b, substr_1, substr_2)$ returns 1 if a appears in the substring $substr_1$ and b appears in the substring $substr_2$, 0 otherwise.

Observe that $\omega_\delta(a, b)$ is actually an overestimation of the true number of matches between a and b in $\langle \bar{s}_1, \bar{s}_2 \rangle_\delta$ because it is not guaranteed that a and b are aligned in $\langle \bar{s}_1, \bar{s}_2 \rangle_\delta$ every time they have been counted as a match.

Armed with the values of ω_δ, we can compute μ_δ as follows. First, we construct a (complete) bipartite weighted graph $G_\delta = (V, U, E, \omega_\delta)$, where each vertex in V (resp. U) is a symbol of Π_1 (resp. Π_2). Then, we compute a *maximum weighted matching* M_δ on G_δ as follows:

$$\mu_\delta = \sum \omega_\delta(a, b) \ s.t. \ (a, b) \in M_\delta$$

Observe that the computation of the maximum weighted matching simulates the construction of the valid matching schema that allows the highest number of matches.

Once these two tasks have been performed, it is possible to state the following proposition:

Proposition 2 $L = \min_{\delta \in [0..l/2]}\{l + \delta - \mu_\delta\}$ is a lower bound for $d^* = \mathcal{L}_{\langle \pi_1, \pi_2, \chi \rangle}(s_1, s_2) =$

$l + \delta^* - \mu^*$, that is $L \leq d^*$.

Proof

To show that $L \leq d^*$, we must quantify the difference between the first and the second term of the inequality.

We start by characterizing μ_δ and we show that it is monotonically increasing. In fact, μ_δ is obtained by counting the co-occurrences of each pair of symbols appearing in a window of size $2 \cdot \delta + 1$, as the substrings considered by κ are $s_2[i-\delta..i+\delta]$ and $s_1[i-\delta..i+\delta]$, respectively. Clearly, enlarging the window will never produce a decrease of the co-occurrences.

Given the value δ^* of δ for the optimal solution, $\mu_{\delta^*} \geq \mu^*$. In fact, we have shown that the estimation of the matchings computed by μ_δ is, actually, an overestimation of the real matchings that can be obtained. Unfortunately, the real value of δ^* cannot be determined without deriving the exact solution.

Now, since $l+\delta^*-\mu^*$ is the minimum value that can be obtained from the exact solution, it follows that $\min_{\delta \in [0..l/2]}\{l + \delta - \mu_\delta\} \leq l + \delta^* - \mu^*$, since $\delta^* \in [0..l/2]$. This proves the proposition. $\square$

Extension to generic values of π_1 and π_2 The philosophy underlying the computation of L for generic values of π_1 and π_2 does not change w.r.t. the case when $\pi_1 = 1$ and $\pi_2 = 1$, examined in detail above. The only difference is that the computation of the maximum weight matching must be substituted by the computation of π-partitions for Π_1 and Π_2 on the bipartite graph such that: *(i)* arcs within the π-partitions are all considered in the computation of μ_δ, and *(ii)* μ_δ is maximum.

Chapter 5

Attacking the giant: heuristic approaches

5.1 Introduction

In Section 4.4 we discussed NP-Hardness of the computation of MPED, where the hardness of finding the optimal matching schema is part of problem. Obviously a naive approach is intractable even for smaller values of parameters such as $|\Pi_1|$ and $|\Pi_2|$. Thus, a heuristic approach is needed. This chapter focuses exactly on the heuristics designed and developed for the computation of the MPED.

We start this chapter by drawing a global view on heuristics in general. Instead of running around in the dark, our research direction was that of implementing various heuristics in order to approach the problem while minimizing the effort. Thus, we decided to make use of a tool called HeuristicLab, that is an environment in which different already implemented heuristics can be used. When we discovered this tool, it sure was surprising and allowed us to concentrate rather on problem design, i.e., how to fit the problem in the settings of HeuristicLab, than on implementing heuristics by ourselves, which surely requires a huge amount of time resource. However, we decided to implement some of the studied heuristics in order to gather a better performance and a interoperable ad-hoc and stand alone version.

Various heuristics, belonging to different families of metaheuristics, have been tested in HeuristicLab and have been compared together. This allowed us to finalize our global view on heuristics and, by using the insights provided by the results, we decided to implement three metaheuristics, namely *(i) Hill Climbing*, *(ii) Simulated Annealing* and *(iii) Evolution Strategy*. The first two have been chosen because they are among the most exploited heuristics in classical optimization problems, and we aimed to check whether a specialized version for MPED could provide good results. They both belong to the family of *local search*, also called *neighbourhood-based search*, and have been used in various contexts and applications. Instead, evolution strategy belongs to the family of *evolutionary algorithms*, where more notable genetic algorithms belong too. Evolution strategy draws inspiration from the biological process of evolution, in which a population evolves through generations. All of these heuristics have been selected for an ad-hoc implementations based on the fact that the concept of matching schema essentially plays an important role. In the first two, the set of matching schemas $\mathbb{M}$ is the set of candidate solutions and for each matching schema $M \in \mathbb{M}$ a neighbourhood can be extrapolated. In the evolution strategy context, $\mathbb{M}$ still represents the set of candidate solutions, and a population, i.e., a subset of $\mathbb{M}$, is considered.

The rest of the chapter is organized as follows: the first global view on heuristics is presented in Section 5.2. Analyzing the results, we pointed out some of the most promising heuristics, thus ad-hoc implementations of hill climbing, simulated annealing and evolution strategy are presented and discussed respectively in Section 5.3.1, Section 5.3.2 and Section 5.4.2. Hill climbing and simulated annealing have been compared together, due to both being neighbourhood-based heuristics; for the evolution strategy, we implemented few variants and compared themselves, also a comparison with hill climbing has been carried out.

Part of the work presented in this chapter has been presented in [23, 25–27, 30, 107].

5.2 A global view on heuristics

Due to the NP-Hardness of the computation of MPED, an approach which exhaustively enumerates every $M \in \mathbb{M}$ is not properly feasible even for small cardinalities of the involved alphabets Π_1 and Π_2. Thus, we strongly rely on using heuristics in order to maintain an acceptable trade-off between the precision of a proposed solution and the time used to compute it.

Initially, a starting point for the design and development of heuristics was that of *neighbourhood-based heuristics*, heuristics which solve computationally hard optimization problem and are based on the concept of a solution and its neighbors, which are solutions themselves. The core approach of a neighbourhood-based heuristic is to explore a so called *search space*, i.e., the set of the canditate solutions for the problem, by starting at some point, i.e., a solution, and then moving to another point by some sort of decisions or criteria. In the case of MPED, the search space for an heuristic is defined by $\mathbb{M}$, where a solution is a matching schema $M \in \mathbb{M}$. A definition of a neighbourhood for a matching schema M is needed. For example, one could define the neighbourhood of M a set N where each $M^i \in N$ differs from M for at most one pair of match between two symbols.

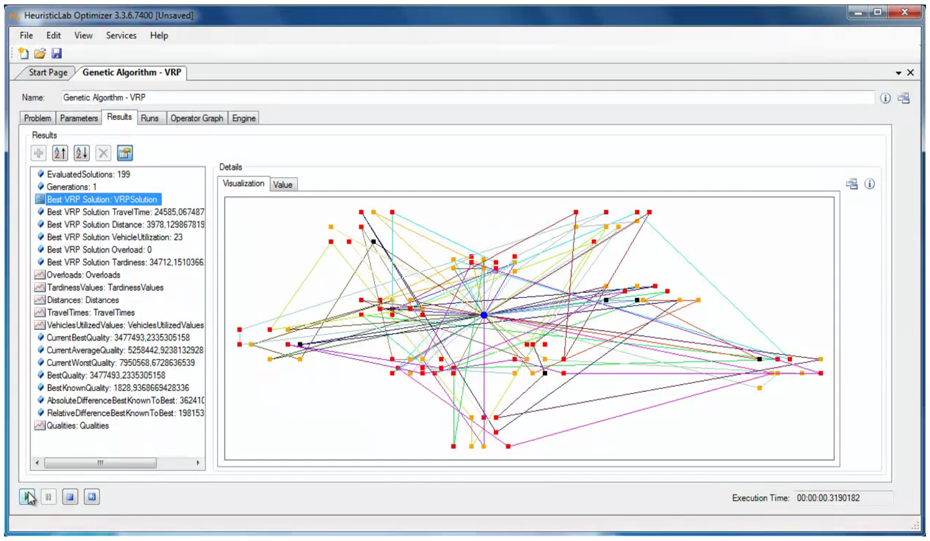
Moreover, there are different families of heuristics which are different from the neighbourhood-based approach. As an example, evolution strategy is a population-based metaheuristic inspired by the natural idea of evolution and adaption. The idea is simple: a population evolves through time by mutation or recombination, best individuals survive while worst are discarded and the best of the best individuals will represent an optimal solution for the problem.

We experimented different heuristics which have been tailored to the problem of computing the MPED, in order to improve the efficiency of heuristic-based computations. For a preliminary view on heuristics, we exploited a tool called HeuristicLab [114], which is a framework providing implementations of different heuristics. We adapted some of the heuristics provided to the computation of MPED and compared them. The rest of this section is dedicated to the aforementioned tool and comparisons.

5.2.1 HeuristicLab

HeuristicLab [114] (HL) is an optimization environment which is intended to provide a general and reliable system for algorithm development, testing and analysis, by allowing users the application of heuristic optimization methods on simple and complex problems. HeuristicLab provides users a graphical user interface where main idea is to represent algorithms as operator graphs which can be modified via drag-and-drop. Main features of HL can be outlined as follows:

- **Various problems and algorithms**: many well-known heuristic algorithms and optimization problems are already defined and implemented in HeuristicLab and can



Name: Genetic Algorithm - VRP

```
EvaluatedSolutions: 199
Generations: 1
Best VRP Solution: VRPSolution
Best VRP Solution TravelTime: 24585,067487
Best VRP Solution Distance: 3978,129867819
Best VRP Solution VehicleUtilization: 23
Best VRP Solution Overload: 0
Best VRP Solution Tardiness: 34712,1510366
Overloads: Overloads
TardinessValues: TardinessValues
Distances: Distances
TravelTimes: TravelTimes
VehiclesUtilizedValues: VehiclesUtilizedValues
CurrentBestQuality: 3477493,2335305158
CurrentAverageQuality: 5258442,9238132928
CurrentWorstQuality: 7950568,6728636539
BestQuality: 3477493,2335305158
BestKnownQuality: 1828,9368669428336
AbsoluteDifferenceBestKnownToBest: 362410
RelativeDifferenceBestKnownToBest: 198153
Qualities: Qualities
```

Execution Time: 00:00:00.3190182

Figure 5-1: Applying a genetic algorithm for a vehicle routing problem. A video tour of HL and further information can be found at https://dev.heuristiclab.com/trac.fcgi/

be used or extended;

- **Visual algorithm designer**: a tool which can be used to model and extend optimization algorithms;
- **Experiment designer and result analysis**: HeuristicLab allows users to define and execute experiments by selecting algorithms, parameters and problems, providing interactive charts for a graphical analysis of results;
- **Parallel and distributed computing**: the support of parallel execution of algorithms both on multi-core or distributed systems is available.

Figure 5-1 represents a screenshot of HL showing a vehicle routing problem.

In our preliminary comparative benchmarks, we tried out various heuristics already present in HL, which are *local search*, *simulated annealing*, *genetic algorithm*, *offspring selection genetic algorithm*, *genetic algorithm with an age-layered population structure*, *evolution strategy* and *offspring selection evolution strategy*. It is due to notice that together with the comparisons of these heuristic, two additional heuristics were used in the experiments, that are *random-restart steepest ascent hill climbing* and *simulated annealing*, both of them implemented in C++.

Selected heuristics are different in the approach, albeit they all share the same goal, which is to find the most optimal $M \in \mathbb{M}$. Following paragraphs expose some details regarding the heuristics. Detailing all of the possibile parameter setups for each heuristics is beyond the scope of this thesis, thus we only briefly indicate core parameters values. The interested reader can refer to [114] for further information.

Local Search Local search heuristic starts at some random starting point in the search space, selects a sample of k neighboring solutions then evaluates those k solutions and moves to the best one. The parameters for a local search heuristics are represented by the *sample size* and *maximum number of iterations*. Table 5.1 shows the values for the specified parameters.

Parameter	Value
Sample Size	$\|\Pi_1\| + \|\Pi_2\|$

Table 5.1: Parameters for Local Search heuristic

Simulated Annealing The simulated annealing [71] implementation available in HeuristicLab is quite configurable as it provides a variety of selectable cooling schedules in the form of different *annealing operators* in addition to parameters for the number of *inner iterations* per step as well as the overall *maximum number of iterations*. A detailed implementation-independent introduction of the simulated annealing heuristic is presented in Section 5.3.2. Table 5.2 shows the values for the specified parameters.

Parameter	Value
Inner Iterations	$\|\Pi_1\| + \|\Pi_2\|$
Start Temperature	100
Annealing Operator	exponential discrete double-value modifier

Table 5.2: Parameters for Simulated Annealing heuristic

Genetic Algorithm A genetic algorithm [85] (GA) is a population-based metaheuristic inspired by natural selection that iteratively evolves a population of solution candidates represented by genomes over a number of *generations* with nature-like operators such as *mutation*, *crossover* and *selection*. In our experiment, we have observed that order-preserving crossover operators work well for our problem, thus we used a partially matched crossover operator in combination with a cyclic crossover for all of the presented GA variants. Some details about mutators (which are in common with Evolution Strategies) are available in Section 5.4.2. Table 5.3 indicates the values for the specified parameters.

Parameter	Value
Population Size	$10(\|\Pi_1\| + \|\Pi_2\|)$
Mutators	swap2, swap3
Crossover Operators	partially matched, cyclic
Elites	1
Mutation Probability	5%
Selector	proportional, with windowing

Table 5.3: Parameters for Genetic Algorithm heuristic

Offspring Selection Genetic Algorithm Offspring selection [1] is an extension to the classical model of a Genetic Algorithm. It adds a selection step after reproduction that truncates low quality crossover results before they ever become part of the new population by comparing the new child's fitness with its parents fitness. The idea is to prevent premature convergence by not drowning out good genes through improper crossover operations and to adaptively steer selection pressure. Offspring Selection GA offers a moderate number of parameters available to setup. The *maximum selection pressure* signals premature conversion of the GA and halts execution if the creation of offspring is failed; the *success ratio* determines what fraction of the offspring has to outperform its parents; the *comparison factor* is used to determine if a child is successful based on the parent's fitness and can dynamic changed ver time. Table 5.4 shows values for the specified parameters.

Genetic Algorithm with an Age-Layered Population Structure A different mechanism to avoid premature convergence can be seen in an Age-Layered Population Structure [61] (ALPS). It works by assign all individuals in the population an age and having individuals compete only within their own layer, but gradually ascend through the layers to more mature individuals as they get older. Additionally, completely new individuals are also generated periodically to replace the ones that aged out of the lowest age layer. This approach aims to give fresh genes a fair chance to gradually mix into the population avoiding them to get eliminated out of the gene-pool immediately. The *number of layers* parameter indicates the number of age groups individual progress through. Table 5.5 shows

Parameter	Value
Population Size	$10(\|\Pi_1\| + \|\Pi_2\|)$
Mutators	swap2, swap3
Crossover Operators	partially matched, cyclic
Elites	1
Mutation Probability	5%
Maximum Selection Pressure	200
Selected Parents	200
Comparison Factor	0
Lower Bound Comparison Factor	1
Upper Bound Success Ratio	1
Offspring Selection Before Mutation	false

Table 5.4: Parameters for Offspring Selection Genetic Algorithm heuristic

values for the specified parameters.

Parameter	Value
Population Size	$10(\|\Pi_1\| + \|\Pi_2\|)$
Mutators	swap2, swap3
Crossover Operators	partially matched, cyclic
Elites	1
Mutation Probability	5%
Number of Layers	10
Selector	generalized rank with pressure 4
Plus Selection	false

Table 5.5: Parameters for Genetic Algorithm with ALPS heuristic

Evolution Strategy Evolution strategies are inspired by evolution and they rely primarily on Mutation and Selection although it is also possible to include crossover operators. As we will notice in Section 5.2.2, the evolution strategy algorithm implemented in HeuristicLab resulted in a very promising approach, thus we decided to implement it as a stand-alone version implemented in C++. A complete discussion on it is presented in Section 5.4.2. The comparisons hereafter available refer to the HeuristicLab version; Table 5.6 shows values for each available parameter.

Offspring Selection Evolution Strategy Offspring selection can be also applied to evolution strategies. Context and properties remain unaltered. Table 5.7 presents value for each available parameter.

Parameter	Value
Population Size	$10(\vert\Pi_1\vert + \vert\Pi_2\vert)$
Mutators	swap2, swap3
Parents Per Child	1
Plus Selection	true

Table 5.6: Parameters for Evolution Strategy algorithm

Parameter	Value
Population Size	$10(\vert\Pi_1\vert + \vert\Pi_2\vert)$
Mutators	swap2, swap3
Max. Selection Pressure	200
Comparison Facto	.5
Success Ratio	1
Elites	1
Selected Parents	40
Plus Selection	true

Table 5.7: Parameters for Offspring Selection Evolution Strategy algorithm

5.2.2 Comparison of different heuristics

Experimental Setup

In this comparison we considered pairs of string which have been generated randomly. The parameters which have been varied are the *alphabets cardinalities* $|\Pi_1|$ and $|\Pi_2|$ and the strings lengths $len(s_1)$ and $len(s_2)$. Moreover, a degree of correlation has been added to each pair of string. Table 5.8 shows the considered values for the parameters.

For each different parameterization one problem instance has been generated based on the selected parameter set. For each problem instance, we applied the heuristics discussed in the previous subsection. To account for the stochasticity of the results, we repeated the execution of each heuristic on the same problem instance 10 times.

Generated results are represented by two types of data visualization components, that are *(i)* line plots of individual runs and *(ii)* tables with the aggregated results for all 10

Parameter	Values to consider
Degree of added correlation	$\{0, .5, 1\}$
Alphabet cardinalities	$\{8, 12, 16\}$
String lengths	256

Table 5.8: Comparative benchmark parameters for HeuristicLab heuristics comparison.

runs per heuristic.

Line plots contain different marked lines for each heuristic. Each line plot represents the first of the 10 runs, thus it is meant to show only an exemplary run for the pair heuristic - parameter configuration. The x-axis indicates the number of evaluations up to the maximum, while the corresponding best obtained MPED value after that number of evaluations is indicated on the y-axis. Therefore, lower y-values are better.

Tables with aggregated results have been generated from each parameter configuration following two steps. In the former step we computed the following values for each problem instance:

- the best obtained MPED value for each heuristic,
- the best obtained MPED value across all heuristics and runs,
- the difference between these two values.

In the latter step we aggregated the results among all 10 problem instances, thus obtaining:

- the average of the best obtained MPED values for each heuristic $E(MPED)$,
- the average of the inaccuracy $E(inacc)$,
- the standard deviation of the obtained MPED values $\sigma(MPED)$.

Standard deviation has been used for the whole populations in this example since we do not want to extrapolate our results beyond our sample with such a small number of runs. The standard deviation of the inaccuracy is the same as the standard deviation of the obtained MPED values, because the individual data points are offset by a constants. Note that discussions of the plots will be slightly presumptive in nature because it is difficult to provide final statements only regarding a survey.

Results

In the following we consider all of the parameters configurations. We start by analyzing the configuration with alphabet cardinality equals to 16, which we believe being the most challenging problem for all the compared heuristics. However, Table 5.9 shows the lowest obtained MPED value among all runs and heuristics.

Alphabet cardinality	Added correlation	Lowest obtained MPED
16	0.0	186
16	0.5	127
16	1.0	0
12	0.0	181
12	0.5	123
12	1.0	0
8	0.0	165
8	0.5	104
8	1.0	0

Table 5.9: Lowest obtained MPED value among all runs and heuristics for HeuristicLab heuristics comparison.

Alphabet cardinality 16 Figure 5-2 shows the graph resulting from the runs with alphabet cardinality 16 and added correlation 0. Clearly, population-based heuristics seem to be performing better in general than the single-solution based heuristic. Note that evolution strategies parameterization is also to be considered population based, albeit it presents smaller populations than genetic algorithms variants. We used this insight in order to design and implement the ad-hoc evolution strategy presented in Section 5.4.2. Table 5.10 shows in detail results obtained by each heuristic. Results for added correlation 0.5 and 1 are respectively represented in Figure 5-3, Figure 5-4, Table 5.11 and Table 5.12.

Alphabet cardinality 12 Decreasing the alphabet cardinality to 12 results in a similar picture across all correlation values. Figure 5-5, Figure 5-6 and Figure 5-7 respectively show graphs resulting from the runs, while Table 5.13, Table 5.14 and Table 5.15 respectively show in detail results obtained by each heuristics. Note that results with no added correlation seem to be even a bit worse than those with alphabet cardinality 16: this is intended because a shorter number of evaluations apparently overtakes the decrease in search space.

Alphabet cardinality 8 Reaching an alphabet size of 8 and considering 0 added correlation seems to decrease the performance gap between population-based heuristics and neighborhood-based heuristics. However, this only happens with 0 added correlation as presumably with a smaller search space it's easier to find a good solution by chance. Adding more correlation, the effect disappears and population-based heuristics clearly gain lead again. Figure 5-8, Figure 5-9 and Figure 5-10 respectively show graphs of the runs, while detailed results obtained by each heuristics are respectively shown in Table 5.16, Table 5.17

and Table 5.18.

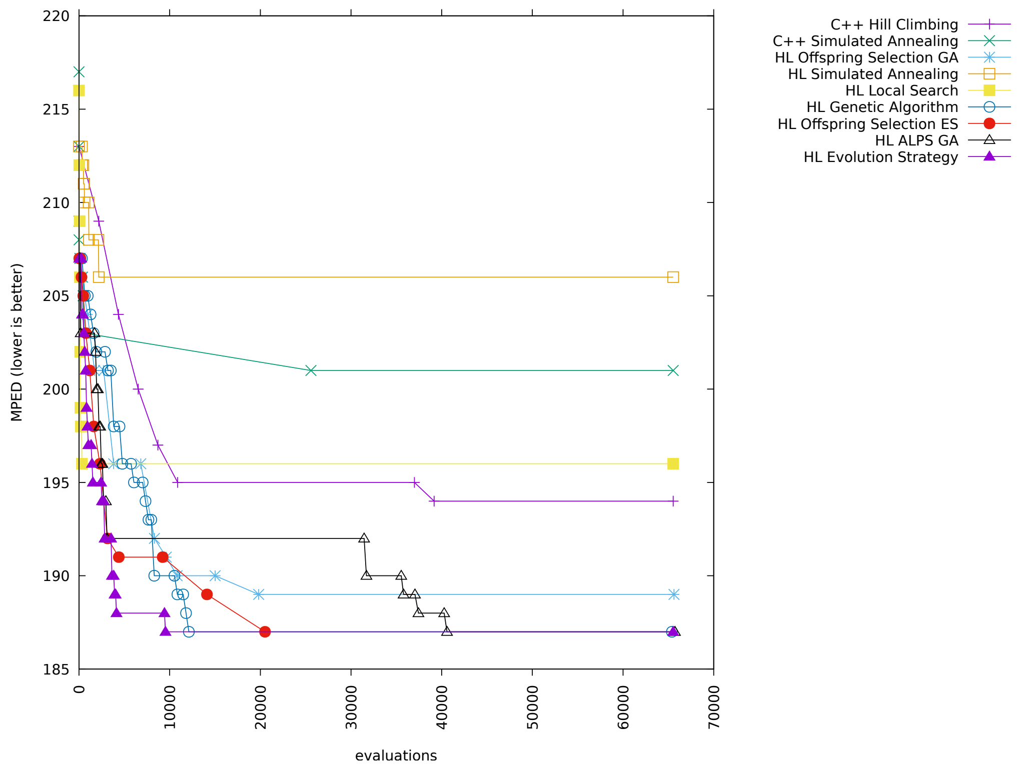


Figure 5-2: Runs with alphabets cardinality 16 and added correlation 0

Heuristic	$E(MPED)$	$E(inacc)$	$\sigma(MPED)$
HL Offspring Selection GA	187.9	1.9	1.3748
HL Genetic Algorithm	188.1	2.1	1.8138
HL Evolution Strategy	189.1	3.1	2.2561
HL Offspring Selection ES	189.1	3.1	1.6401
HL ALPS GA	189.2	3.2	1.4697
C++ Hill Climbing	193.5	7.5	1.8028
HL Local Search	195.2	9.2	2.7857
C++ Simulated Annealing	198.8	12.8	1.7205
HL Simulated Annealing	208.8	22.8	2.8213

Table 5.10: HL heuristics comparison, alphabet cardinality 16 and added correlation 0

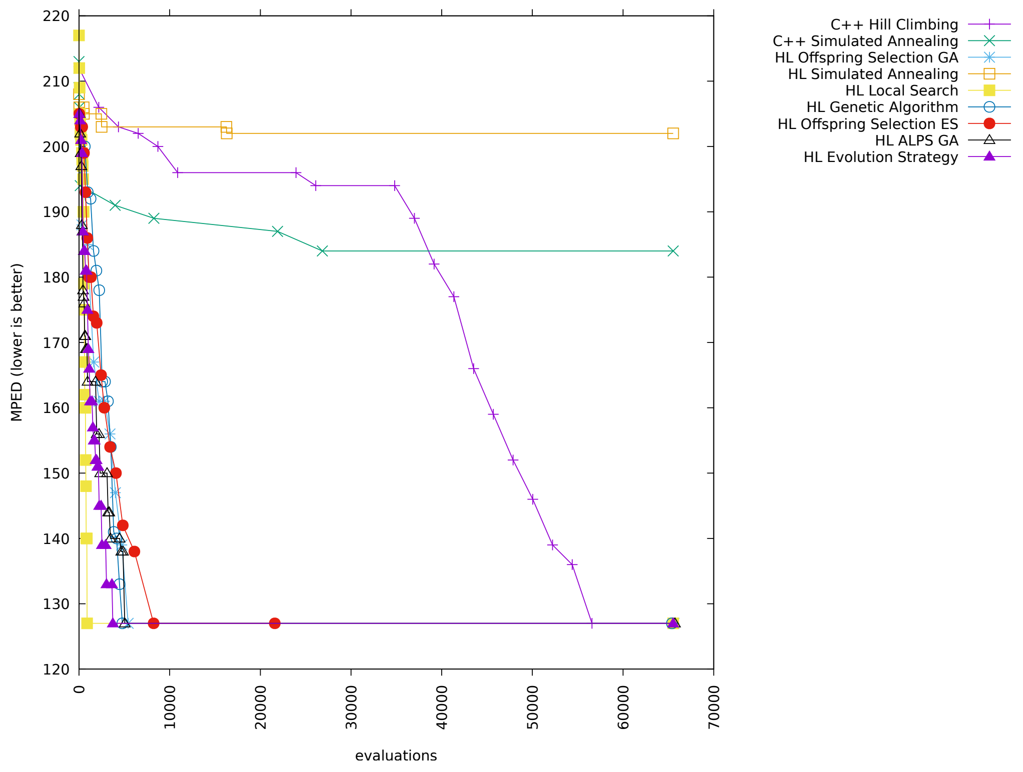


Figure 5-3: Runs with alphabets cardinality 16 and added correlation 0.5

Heuristic	$E(MPED)$	$E(inacc)$	$\sigma(MPED)$
HL ALPS GA	127	0.0	0.0000
HL Evolution Strategy	127	0.0	0.0000
HL Genetic Algorithm	127.0	0.0	0.0000
HL Offspring Selection ES	127	0.0	0.0000
HL Offspring Selection GA	127	0.0	0.0000
C++ Hill Climbing	128.3	1.3	3.9000
HL Local Search	148	21.0	32.0905
C++ Simulated Annealing	180.1	53.1	2.4678
HL Simulated Annealing	206.9	79.9	4.1340

Table 5.11: HL heuristics comparison, alphabet cardinality 16 and added correlation 0.5

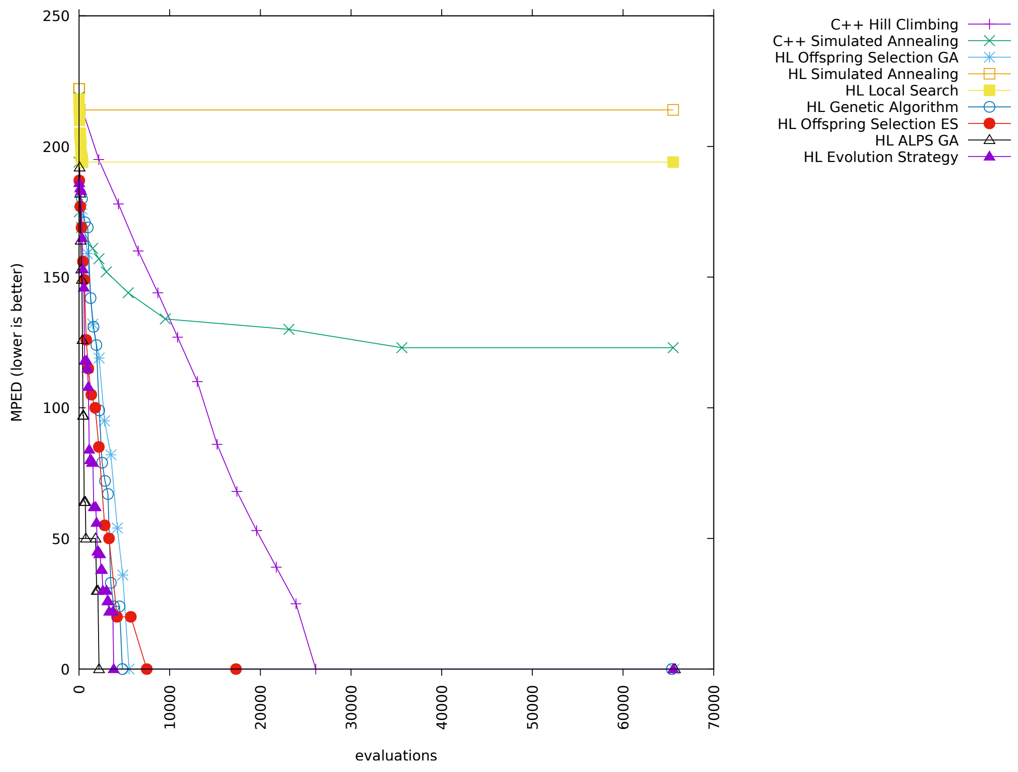


Figure 5-4: Runs with alphabets cardinality 16 and added correlation 1

Heuristic	$E(MPED)$	$E(inacc)$	$\sigma(MPED)$
C++ Hill Climbing	0	0.0	0.0000
HL ALPS GA	0	0.0	0.0000
HL Evolution Strategy	0	0.0	0.0000
HL Genetic Algorithm	0	0.0	0.0000
HL Offspring Selection ES	0	0.0	0.0000
HL Offspring Selection GA	0	0.0	0.0000
HL Local Search	58.2	58.2	88.9065
C++ Simulated Annealing	127.9	127.9	8.2879
HL Simulated Annealing	202.7	202.7	8.1615

Table 5.12: HL heuristics comparison, alphabet cardinality 16 and added correlation 1

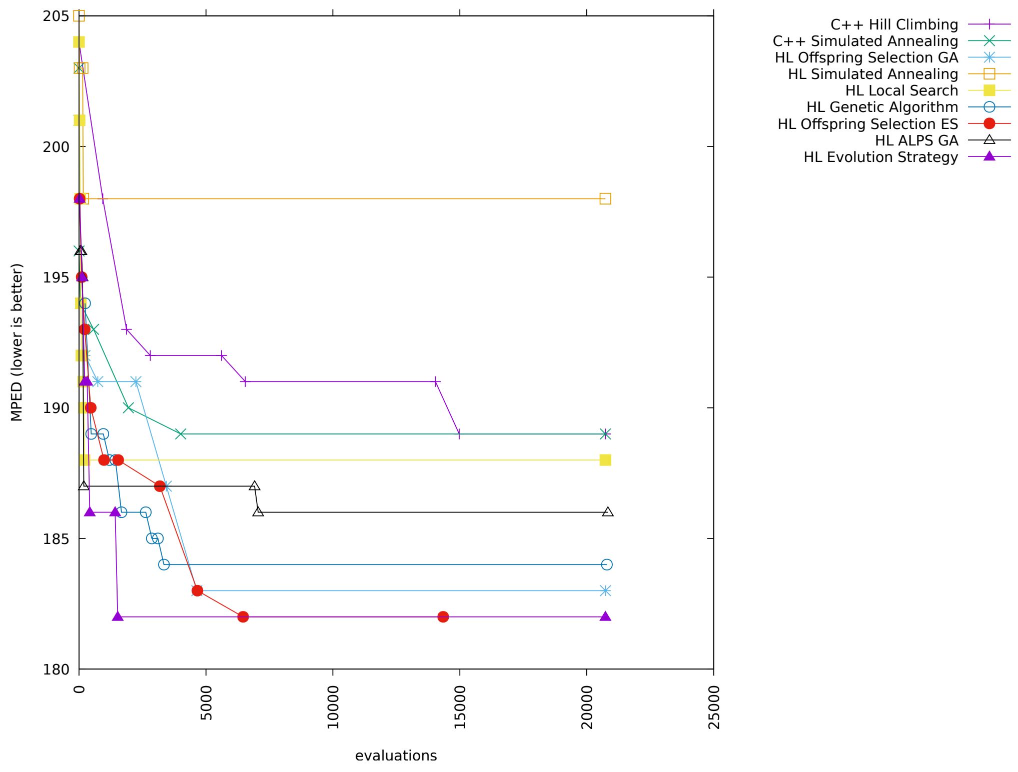


Figure 5-5: Runs with alphabets cardinality 12 and added correlation 0

Heuristic	$E(MPED)$	$E(inacc)$	$\sigma(MPED)$
HL Offspring Selection ES	183.2	2.2	1.2490
HL Offspring Selection GA	183.3	2.3	1.0050
HL Evolution Strategy	183.4	2.4	1.2000
HL Genetic Algorithm	183.7	2.7	1.4177
HL ALPS GA	183.8	2.8	1.4697
C++ Hill Climbing	186.9	5.9	1.9209
C++ Simulated Annealing	189.1	8.1	0.7000
HL Local Search	189.3	8.3	2.7946
HL Simulated Annealing	196.3	15.3	1.7349

Table 5.13: HL heuristics comparison, alphabet cardinality 12 and added correlation 0

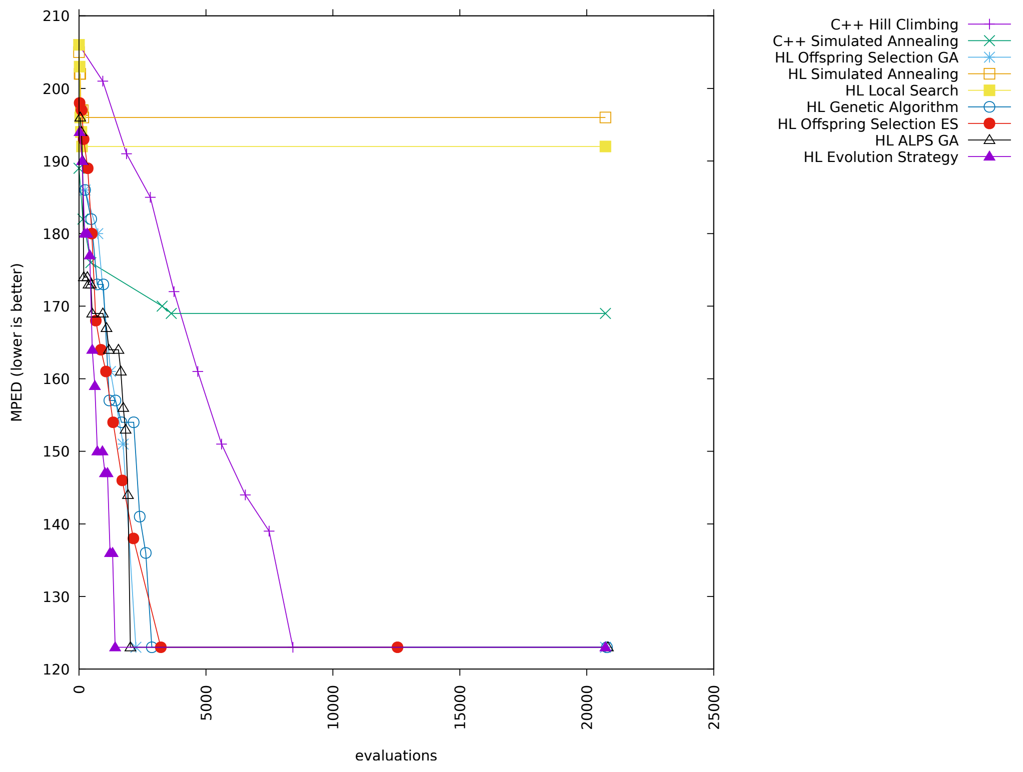


Figure 5-6: Runs with alphabets cardinality 12 and added correlation 0.5

Heuristic	$E(MPED)$	$E(inacc)$	$\sigma(MPED)$
C++ Hill Climbing	123	0.0	0.0000
HL ALPS GA	123	0.0	0.0000
HL Evolution Strategy	123	0.0	0.0000
HL Genetic Algorithm	123.0	0.0	0.0000
HL Offspring Selection ES	123	0.0	0.0000
HL Offspring Selection GA	123	0.0	0.0000
HL Local Search	156.8	33.8	33.8757
C++ Simulated Annealing	165.3	42.3	4.4283
HL Simulated Annealing	195.6	72.6	3.6661

Table 5.14: HL heuristics comparison, alphabet cardinality 12 and added correlation 0.5

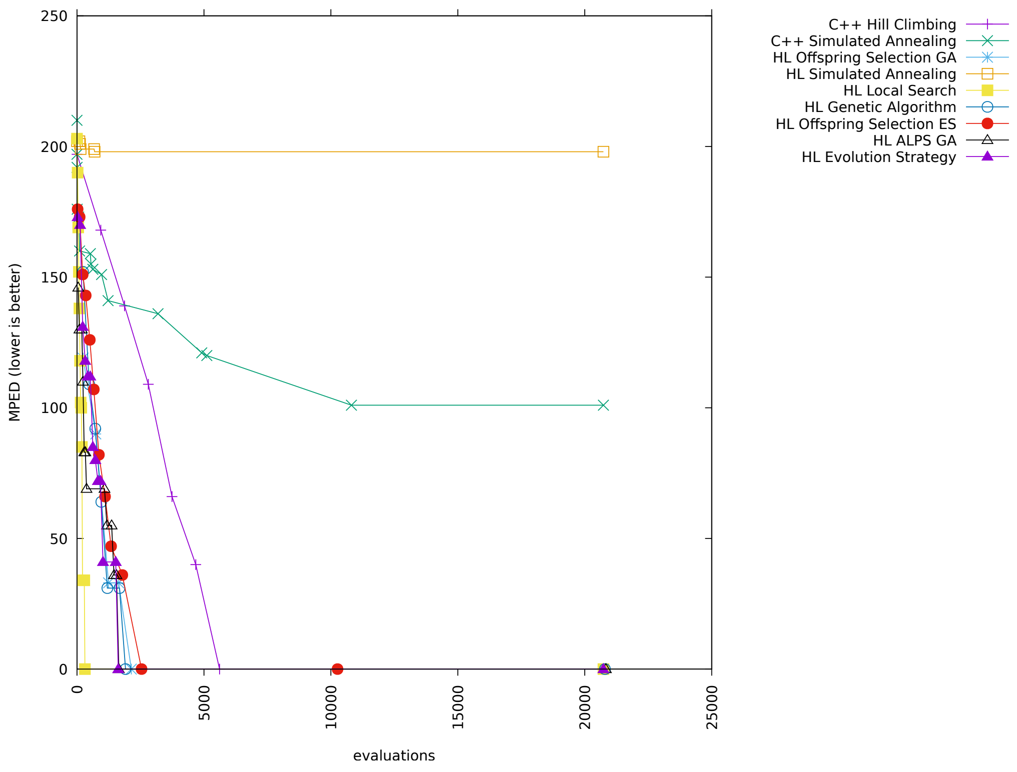


Figure 5-7: Runs with alphabets cardinality 12 and added correlation 1

Heuristic	$E(MPED)$	$E(inacc)$	$\sigma(MPED)$
C++ Hill Climbing	0	0.0	0.0000
HL ALPS GA	0	0.0	0.0000
HL Evolution Strategy	0	0.0	0.0000
HL Genetic Algorithm	0	0.0	0.0000
HL Local Search	0	0.0	0.0000
HL Offspring Selection ES	0	0.0	0.0000
HL Offspring Selection GA	0	0.0	0.0000
C++ Simulated Annealing	97.7	97.7	8.9336
HL Simulated Annealing	188.2	188.2	12.1885

Table 5.15: HL heuristics comparison, alphabet cardinality 12 and added correlation 1

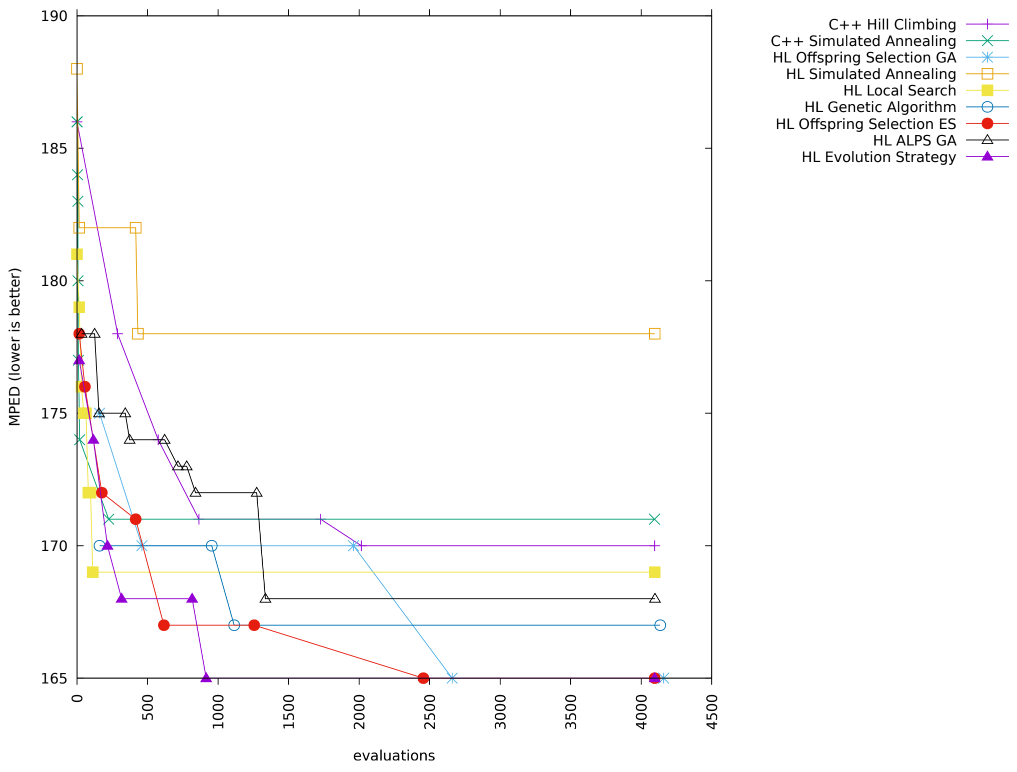


Figure 5-8: Runs with alphabets cardinality 8 and added correlation 0

Heuristic	$E(MPED)$	$E(inacc)$	$\sigma(MPED)$
HL Offspring Selection GA	165.8	0.8	1.2490
HL Offspring Selection ES	166.1	1.1	1.1358
HL Genetic Algorithm	166.6	1.6	1.1136
HL ALPS GA	166.8	1.8	1.7205
HL Evolution Strategy	166.9	1.9	1.4457
C++ Simulated Annealing	169.2	4.2	1.9391
C++ Hill Climbing	169.4	4.4	1.3565
HL Local Search	170.6	5.6	2.3749
HL Simulated Annealing	176.1	11.1	3.2078

Table 5.16: HL heuristics comparison, alphabet cardinality 8 and added correlation 0

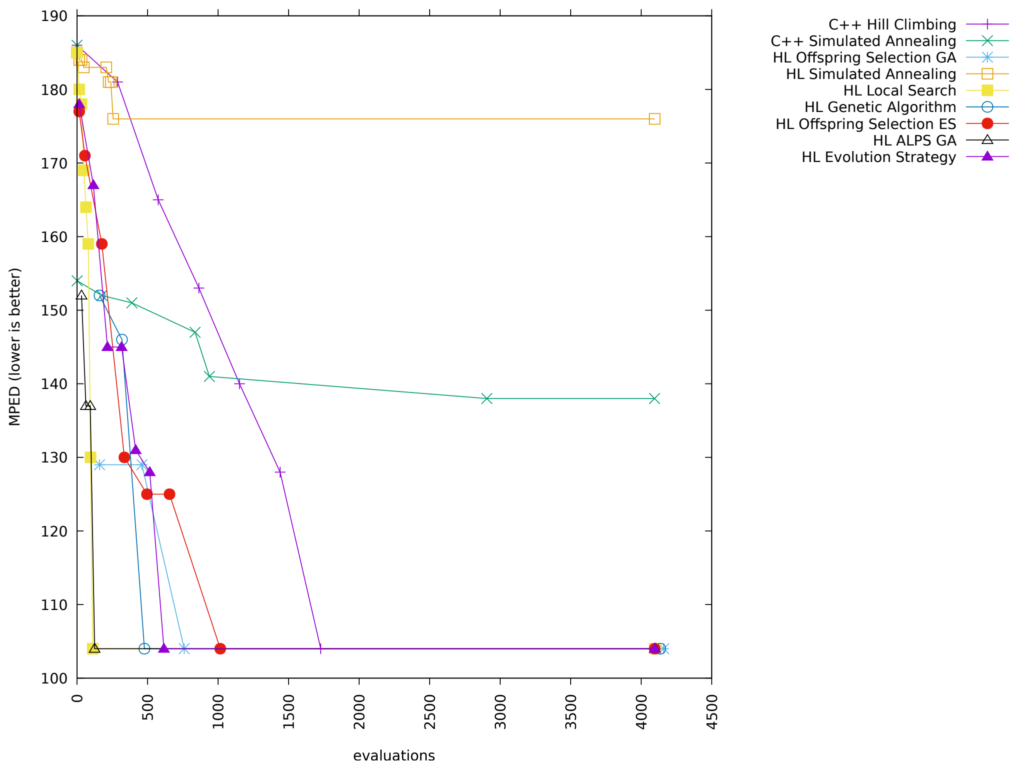


Figure 5-9: Runs with alphabets cardinality 8 and added correlation 0.5

Heuristic	$E(MPED)$	$E(inacc)$	$\sigma(MPED)$
C++ Hill Climbing	104	0.0	0.0000
HL ALPS GA	104	0.0	0.0000
HL Evolution Strategy	104	0.0	0.0000
HL Genetic Algorithm	104	0.0	0.0000
HL Offspring Selection ES	104	0.0	0.0000
HL Offspring Selection GA	104	0.0	0.0000
HL Local Search	110.6	6.6	19.8000
C++ Simulated Annealing	128.8	24.8	12.9754
HL Simulated Annealing	170.9	66.9	6.8768

Table 5.17: HL heuristics comparison, alphabet cardinality 8 and added correlation 0.5

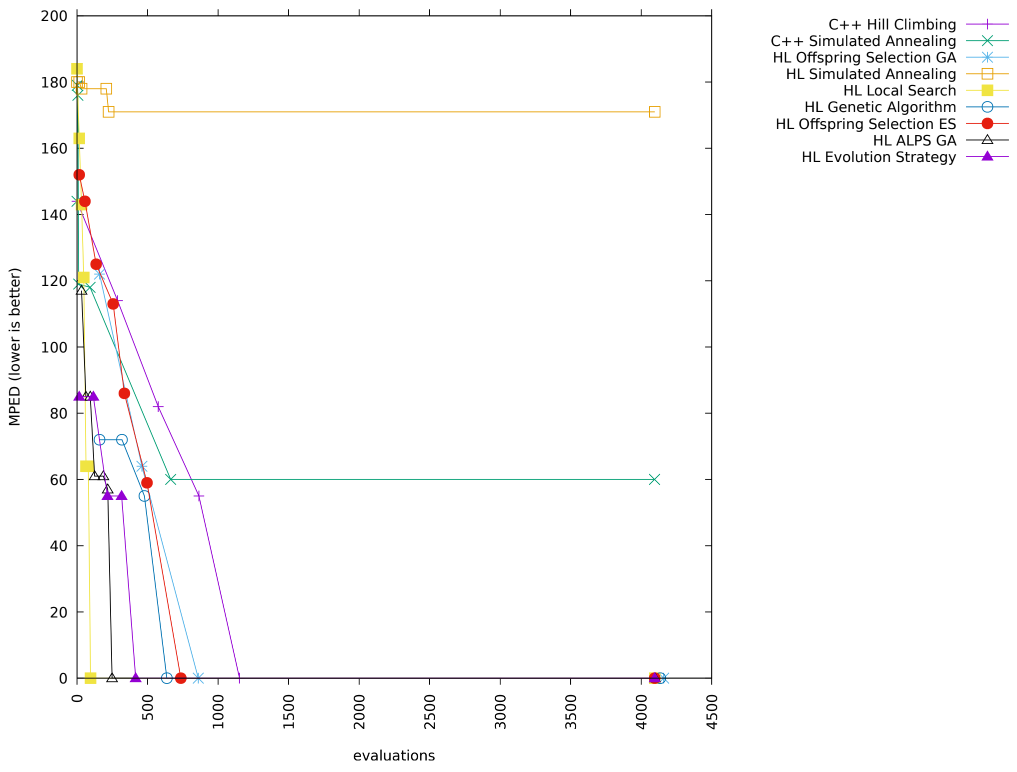


Figure 5-10: Runs with alphabets cardinality 8 and added correlation 1

Heuristic	$E(MPED)$	$E(inacc)$	$\sigma(MPED)$
C++ Hill Climbing	0	0.0	0.0000
HL ALPS GA	0	0.0	0.0000
HL Evolution Strategy	0	0.0	0.0000
HL Genetic Algorithm	0	0.0	0.0000
HL Local Search	0	0.0	0.0000
HL Offspring Selection ES	0	0.0	0.0000
HL Offspring Selection GA	0	0.0	0.0000
C++ Simulated Annealing	49.4	49.4	24.9648
HL Simulated Annealing	150.8	150.8	23.6761

Table 5.18: HL heuristics comparison, alphabet cardinality 8 and added correlation 1

5.3 Survey on local search heuristics

5.3.1 Hill Climbing

The basic idea underlying the proposed heuristic is that, if we were armed with the best matching schema, then the computation of MPED between s_1 and s_2 could be done easily by means of dynamic programming.

Hill climbing heuristic works as follows: intuitively, at step 0, a starting matching schema M^0 is chosen, and $\mathcal{L}^{M^0}(s_1, s_2)$ is computed. At the generic iteration i, the neighbors $M^i_{\nu_j}$ of the current matching schema M^i are considered, and the distances $\mathcal{L}^{M^i_{\nu_j}}(s_1, s_2)$ are computed. A neighbor $M^i_{\nu_j}$ of a matching schema M^i is a perturbation of M^i that exchanges only one pair of symbols between two partitions of the same alphabet.

The matching schema guaranteeing the lowest distance is, then, chosen and set as the starting matching schema M^{i+1} for the next step. This activity stops when the edit distance cannot be further improved. When this happens, the current edit distance is returned as result. To avoid repeated computations, a hash map registering the already verified matching schemas is exploited.

Finally, to increase the chances of finding the optimal alignment, a certain number of random restarts, each characterized by a new randomly selected matching schema, are carried out.

Since, at each step, we are interested to find a matching schema returning a distance $\mathcal{L}^{M^i_{\nu_j}}(s_1, s_2)$ lower than the current minimum d_{min}, for computing $\mathcal{L}^{M^i_{\nu_j}}(s_1, s_2)$ we resort to the approach presented by Landau and Vishkin in [73], which is able to compute the edit distance between s_1 and s_2 in $O(d_{min} \cdot max\{len(s_1), len(s_2)\})$, if this distance is less than d_{min}.

The ideas illustrated above are formalized in the Algorithm `HILL-CLIMBING`, reported in Algorithm 1.

We observe that explicitly storing, comparing, and ordering matching schemas can be computationally heavy tasks, from both the execution time and the space occupancy perspectives. In our solution, we take advantage of some peculiarities of matching schemas, which allow us to resort to a *virtual and compact* representation of them. This allows significant reductions of both the time and the space required to handle matching schema manipulation.

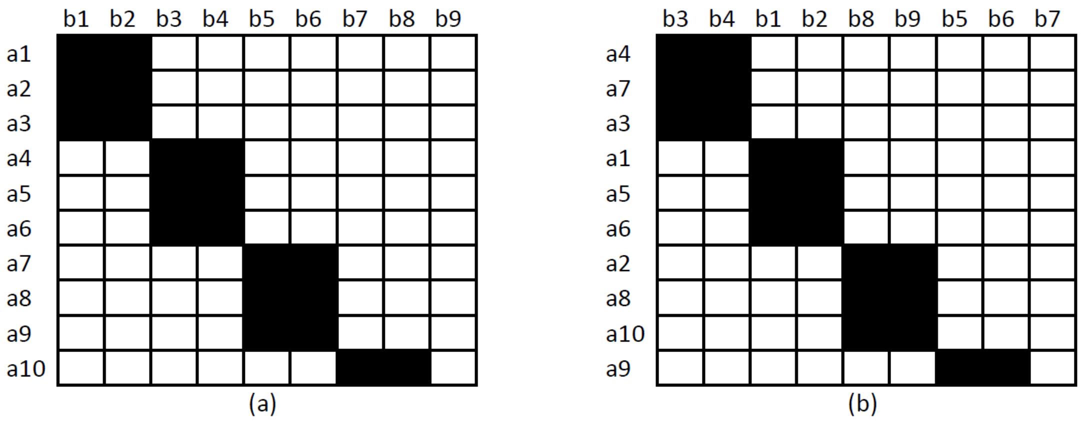

b1 b2 b3 b4 b5 b6 b7 b8 b9
a1 a2 a3 a4 a5 a6 a7 a8 a9 a10
(a)
b3 b4 b1 b2 b8 b9 b5 b6 b7
a4 a7 a3 a1 a5 a6 a2 a8 a10 a9
(b)

Figure 5-11: Examples of $\langle \pi_1, \pi_2 \rangle$-matching schemas for $\pi_1 = 3$ and $\pi_2 = 2$

In particular, given the pair $\langle \pi_1, \pi_2 \rangle$, a valid configuration of a $\langle \pi_1, \pi_2 \rangle$-matching schema has always the form depicted in Figure 5-11(a) – here a $\langle 3, 2 \rangle$-matching schema is used as an example. Observe that, if we change the order of the symbols in Π_1 and Π_2, instead of the content of the matrix, as represented in Figure 5-11(b), the representation of the matching schema still holds. As a consequence, a cheaper representation of matching schemas may simply resort to the order of the symbols in Π_1 and Π_2, as they implicitly imply partitions, matches, and mismatches. For instance, the matching schema of Figure 5-11(b) can be simply represented by the pair of ordered symbols $\Pi_1 = \{a4, a7, a3, a1, a5, a6, a2, a8, a10, a9\}$ and $\Pi_2 = \{b3, b4, b1, b2, b8, b9, b5, b6, b7\}$; in this case, partitions $\{a4, a7, a3\}$ of Π_1 and $\{b3, b4\}$ of Π_2 match. With this representation at hand, changing the matchings of a symbol a_i in Π_1 with symbols in Π_2 simply translates with a change in the position of a_i in Π_1.

In our solution, we exploit this representation for initializing matching schemas, for computing neighbors and for storing checked matching schemas. Since this representation is transparent to the actual usage of matching schemas, for the sake of presentation, we express operations on matching schemas as they were explicitly represented as matrices.

One of the most interesting operations carried out in `HILL-CLIMBING` is the construction of the neighbors of a given matching schema M. This task is formalized in the function `Neighbours`, reported in Algorithm 2. In particular, given a starting matching schema M, its rows and columns are virtually swapped one by one to generate matching schemas that differ from M for only one symbol swap in partitions.

```
Input  : two strings $s_1$ and $s_2$ over the alphabets $\Pi_1$ and $\Pi_2$, respectively;
         a set $\chi$ of constraints;
         three integers $\pi_1$, $\pi_2$, and $T$;
Output: $\mathcal{L}_{\langle\pi_1,\pi_2,\chi\rangle}(s_1, s_2)$;
Data   : two $|\Pi_1| \times |\Pi_2|$ matching schemas $M$ and $M'$;
         Checked: a hash map for tested matching schemas;
         improved: boolean;
         $t$, mindist, globaldist: integer
begin
    $t = 0$;
    initialize($M$,$\pi_1$,$\pi_2$,$\chi$);
    mindist = $\mathcal{L}^M(s_1, s_2)$;
    globaldist = mindist;
    improved = true;
    Checked = $\emptyset$;
    while improved do
        improved = false;
        $N$ = Neighbours($M, \chi$);
        foreach $M^{'}$ in $N$ and not in Checked do
            Checked = Checked $\cup$ $M^{'}$;
            if $\mathcal{L}^{M^{'}}(s_1, s_2) <$ mindist then
                mindist = $\mathcal{L}^{M^{'}}(s_1, s_2)$;
                improved = true;
                $M = M^{'}$;
            end
        end
        if not improved then
            if mindist < globaldist then
                globaldist = mindist;
                improved = true;
                $t = 0$;
            else if $t < T$ then
                $t = t + 1$;
                improved = true;
                $M$ = randomSelect($M$,$\pi_1$,$\pi_2$,$\chi$);
                mindist = $\mathcal{L}^M(s_1, s_2)$;
            end
        end
    end
    return globaldist;
end
```

Algorithm 1: The algorithm HILL-CLIMBING for the computation of $\mathcal{L}_{\langle\pi_1,\pi_2,\chi\rangle}(s_1, s_2)$.

```
Function Neighbours ($M, \chi$)
    Input  : a matching schema $M$ over the alphabets $\Pi_1$ and $\Pi_2$, respectively;
             a set of constraints $\chi$;
    Output: a set of matching schemas $MS$;
    Data   : $n$, $m$, $r_1$, $r_2$, $c_1$, $c_2$: integer;
    begin
        $MS = \emptyset$;
        for $r_1 = 0 \to |\Pi_1|$ do
            for $r_2 = r_1 \to |\Pi_1|$ do
                swaprows($M$, $r_1$, $r_2$);
                for $c_1 = 0 \to |\Pi_2|$ do
                    for $c_2 = c_1 \to |\Pi_2|$ do
                        swapcolumns($M$, $c_1$, $c_2$);
                        if isvalid($M$,$\chi$) then
                            $MS = MS \cup M$;
                    end
                end
            end
        end
        return $MS$;
    end
```

Algorithm 2: The function `Neighbours` for the construction of the neighbors of a given matching schema M.

5.3.2 Simulated Annealing

In order to span over local search heuristics, we implemented also an ad-hoc solution based on Simulated Annealing (SA) [71]. It is a probabilistic metaheuristic for the generic optimization problem of locating a good approximation to the global optimum of a given function in a large search space.

The goal of SA is to bring a system, from an arbitrary initial state, to a state with the minimum possible energy; in our context, the initial state is a valid matching schema, and the state with the minimum possible energy is the one corresponding to an optimal matching schema. Each step of SA consists in selecting a neighbor state n' of the current state n and deciding whether to move the system to n' or stay in n; in our case, each state is a matching schema, and the neighbor of a state is a random perturbation of the current matching schema. A control parameter required by SA is called *temperature*, which determines both the number of iterations that are carried out by the algorithm and the behavior of the random walk. In our approach we empirically set it to $(len(s_1) + len(s_2))(|\Pi_1| + |\Pi_2|)$.

Figure 3 shows the pseudocode of SA. M is initialized with a valid starting assignment. Until the temperature decreases, a random neighbor M' is selected and the distance with this matching schema is computed. The **if** at line 9 of the algorithm decides whether to move to the new state of the system or not. This is certainly done if the `current` distance is worse than the `next` one; otherwise, a probability function [84] is used, indicated here as $accept()$. This function is based on both the *energy* of the current and the next states and on the value of current temperature. In case of a state change, the current matching schema M becomes $M^{'}$ and the `best` distance is updated as well. The functions *initialize* and *randomNeighbors* generate random valid matching schemas.

The output of the function is the *distance*, i.e. the dissimilarity, between s_1 and s_2.

5.3.3 Experimental Analysis

The purpose of this part of the thesis is to evaluate the applicability of the proposed framework and the effectiveness of the proposed neighborhood-based heuristics (herefter HC and SA) for computing $\mathcal{L}_{\langle \pi_1, \pi_2, \chi \rangle}(s_1, s_2)$.

In the following experiments, we considered two main aspects, namely *(i)* the reliability and *(ii)* the execution time of the two heuristics into consideration. In order to precisely

```
Input  : two strings $s_1$ and $s_2$ over the alphabets $\Pi_1$ and $\Pi_2$, respectively;
         a set $\chi$ of constraints;
         two integers $\pi_1$ and $\pi_2$
Output: $\mathcal{L}_{\langle\pi_1,\pi_2,\chi\rangle}(s_1, s_2)$
Data   : $T$, current, next, best: integers
begin
    $T = (len(s_1) + len(s_2))(|\Pi_1| + |\Pi_2|)$;
    $initialize(M,\pi_1,\pi_2,\chi)$;
    current = $\mathcal{L}^M(s_1, s_2)$;
    while $T > 0$ do
        $T = T - 1$;
        $M' = randomNeighbor(M, \pi_1, \pi_2, \chi)$;
        next = $\mathcal{L}^{M'}(s_1, s_2)$;
        if current > next or accept() then
            $M = M'$;
            current = next;
            if current < best then
                best = current;
            end
        end
    end
    return best;
end
```

Algorithm 3: Simulated annealing (SA) pseudocode

evaluate the reliability of the two heuristics, we compared the results returned by them on a set of test data, pre-labeled with the exact value of $\mathcal{L}_{\langle\pi_1,\pi_2,\chi\rangle}(s_1, s_2)$ for them, computed by means of an exhaustive approach (hereafter EX where needed). This approach explores all possible matching schemas and computes the lowest edit distance. As pointed out in Section 4.4, EX is unfeasible for large values of $|\Pi_i|$ and π_i, thus this test was carried out only for small data sets. The analysis of execution times has the dual purpose of both measuring the gain of a heuristic w.r.t the exact solution and verifying its actual applicability to real cases.

The next part is structured as follow: first, we describe the exploited dataset, then we discuss the results of our evaluations on reliability and execution times.

Dataset

The dataset for this experiment has been generated along the lines of the previous ones. Each element of the dataset is a pair of strings (s_1, s_2); both s_1 and s_2 are randomly generated. Without loss of generality, for each instance we randomly generated the strings assuming the following conditions:

- $len(s_1) = len(s_2) = len(s)$,
- $|\Pi_1| = |\Pi_2| = |\Pi|$,
- $\pi_1 = \pi_2 = \pi_i$.

With these conditions, we generated a dataset with the following parameters:

- $len(s) = \{50, 100, 200, 350, 500\}$,
- $|\Pi| = \{3..10\}$,
- $\pi_i = \{1..4\}$ and such that $\pi_i < \lceil |\Pi|/2 \rceil$.

Reliability Analysis

As previously pointed out, we compared the results provided by the two heuristics, HC and SA, with those returned by EX. It was possible to obtain the exact results in a reasonable amount of time only for some of the instances available in the dataset. For each test, we carried out ten executions of the heuristics and averaged the obtained solutions.

Precision of the solutions provided by HC has been measured as follow: let d_{EX} be the exact solution for an instance of $\mathcal{L}_{\langle \pi_1, \pi_2, \chi \rangle}(s_1, s_2)$and let d_{HC} be the average solution returned by HC. The precision P_{HC} thus is computed as

$$P_{HC} = 1 - \frac{d_{HC}}{d_{EX}}; \tag{5.1}$$

P_{SA}, i.e., the precision of SA, was computed analogously. Note that is always that $d_{EX} > 0$.

Table 5.19 illustrates P_{HC} and P_{SA} for some test data, in particular, for each combination of $len(s)$, $|\Pi|$ and π_i. Note that a precision of 1.00 means that the corresponding heuristics provided exactly the value d_{EX}. N/A in the table stands for instances where an excessive execution time was required by EX and, consequently, it was not possible to compute P_{HC} and P_{SA}.

		$len(s)$									
		50		100		200		350		500	
π	$\|\Pi\|$	P_{HC}	P_{SA}	P_{HC}	P_{SA}	P_{HC}	P_{SA}	P_{HC}	P_{SA}	P_{HC}	P_{SA}
1	3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	4	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.99	1.00	1.00
	5	0.98	0.98	0.96	0.96	0.99	0.99	0.99	0.99	0.99	0.99
	6	1.00	1.00	0.97	0.97	1.00	1.00	0.98	0.98	0.97	0.97
	7	1.00	1.00	0.97	0.97	0.96	0.96	1.00	1.00	0.97	0.97
	8	0.93	0.93	0.98	0.98	0.96	0.96	0.97	0.97	1.00	0.98
	9	1.00	0.93	1.00	1.00	1.00	0.98	0.98	0.98	0.99	0.99
	10	0.93	0.93	0.95	0.95	0.97	0.97	1.00	0.97	0.97	0.97
2	5	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.99
	6	1.00	1.00	1.00	1.00	0.96	0.96	0.99	0.99	0.97	0.97
	7	1.00	1.00	0.98	0.98	0.97	0.97	N/A	N/A	N/A	N/A
3	7	0.93	0.93	0.94	0.94	1.00	1.00	N/A	N/A	N/A	N/A

Table 5.19: Values of P_{HC} (left value) and P_{SA} (right value) for different values of π, $|\Pi|$ and $len(s)$. N/A represents an instance for which the exact value cannot be computed in a feasible time.

It is interesting to observe that, even with long sequences or large alphabets, P_{HC} and P_{SA} are always very high. This is encouraging in expecting a very high reliability of the two heuristics in real application cases.

Execution Time Analysis

To evaluate the temporal behavior of HC and SA we designed two main experiments focusing on the variations of $len(s)$ and $|\Pi|$, respectively.

The first experiment is centered on the variation of $|\Pi|$. For each value of $|\Pi|$ belonging to the integer interval $[3, 10]$, we computed the running time necessary to obtain $\mathcal{L}_{\langle 1,1,\emptyset\rangle}$ by applying EX, on the one hand, and HC and SA, on the other hand. The same computation has been carried out for each value of $len(s)$ belonging to the set $\{100, 200, 350, 500\}$. Figure 5-12 shows obtained results. Here, for each value of $len(s)$, we show the increase of the execution time against $|\Pi|$. As expected, analyzing this figure shows that EX has an exponential increase of its running time. We see that SA is always slower than HC and, in several cases, even slower than EX. This is due to the intrinsic nature of this heuristic, which may require a certain number of unnecessary iterations in some cases. Finally, HC is always faster than EX and always below 1 second within the tested dataset.

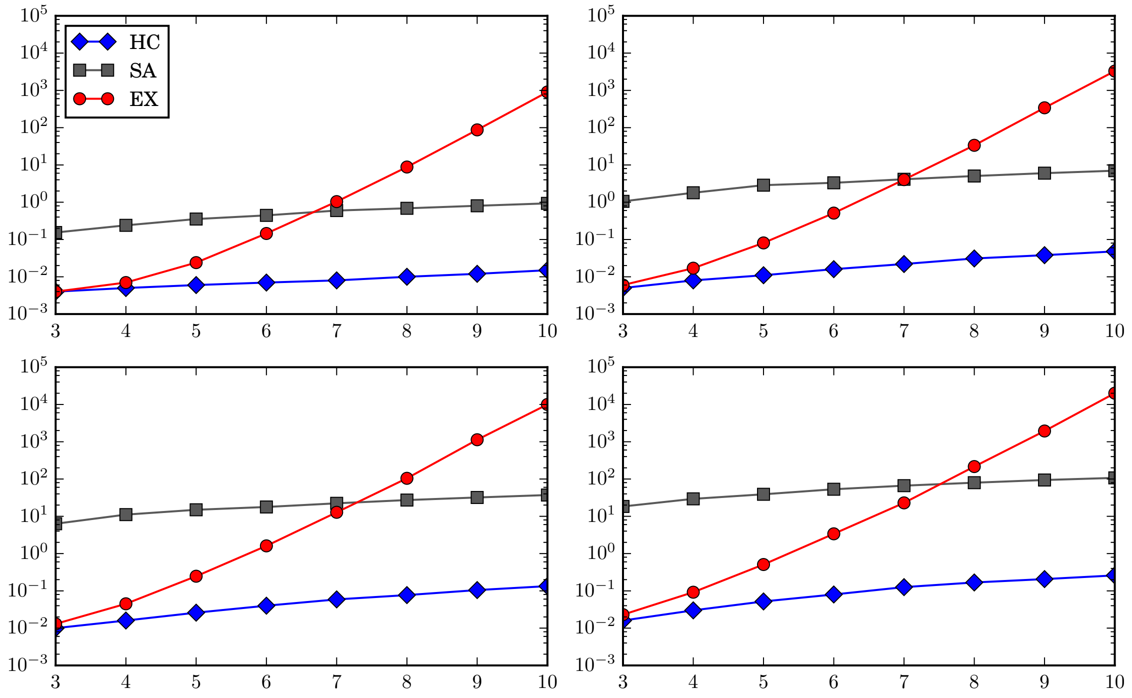


Figure 5-12: Execution time (in seconds) of HC, SA and EX against $|\Pi|$. The four graphs, from top to bottom and from left to right, indicate the results for $len(s)$ = 100, 200, 350 and 500, respectively.

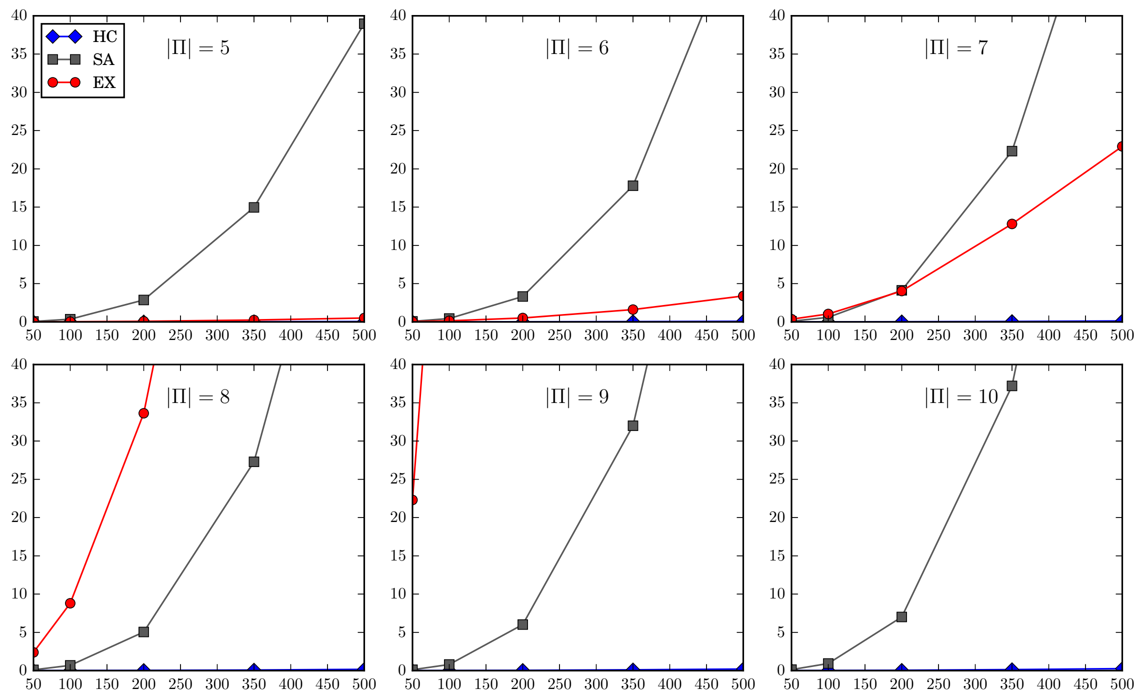


Figure 5-13: Execution time (in seconds) of HC, SA and EX against $len(s)$. The six graphs, from top to bottom and from left to right, indicate the result for $|\Pi| \in \{5..10\}$.

The second experiment focuses on the variation of the execution time of EX, HC and SA against $len(s)$. In particular, for each value of $len(s)$ belonging to the set $\{50, 100, 200, 350, 500\}$, we computed the running time necessary to obtain $\mathcal{L}_{\langle 1,1,\emptyset\rangle}(s_1, s_2)$ for EX, HC, and SA. In this case, we considered six different values of $|\Pi|$, i.e., $|\Pi| \in \{5..10\}$. Therefore, we obtained six different graphs. Results are shown in Figure 5-13.

From the analysis of these graphs, it is easy to see that an increase of l and an increase of $|\Pi|$ lead to a strong increase of the execution time of EX, which becomes out of scale already for $len(s) = 50$ with $|\Pi| = 10$. The execution time of HC is basically flat and always below 1 second for each configuration; this makes HC far more performing than SA.

These last two experiments, along with the essentially identical reliability of HC and SA, make HC the preferred local search heuristic.

As a final test, we analyzed the execution time of `HC` against $len(s)$ for several values of $|\Pi|$. Figure 5-14 shows obtained results. The analysis of this figure confirms the quadratic dependency of the execution time from $len(s)$, caused by the dynamic programming solution underlying it. The dependency from $|\Pi|$ is small, both because the values of $|\Pi|$ are usually

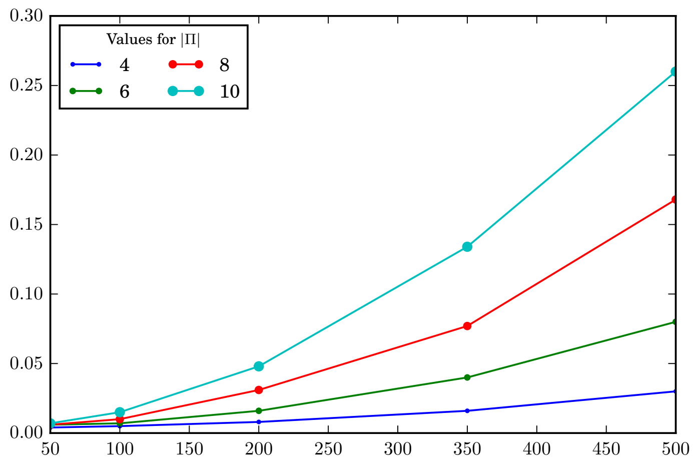


Figure 5-14: Execution time (in seconds) of `HC` against $len(s)$ for several values of $|\Pi|$.

much smaller than $len(s)$ and because our way of computing neighbors for matching schemas allows `HC` to avoid the necessity to consider a great number of matching schemas.

5.4 Survey on evolution strategies

Evolution Strategy, also called Evolutionary Strategy or Evolutionary Programming is a population-based metaheuristic [8, 42, 103]. It belongs to the generic family of evolutionary algorithms, which include, for instance, the more famous Genetic Algorithm. Evolutionary algorithms, thus evolution strategy too, are inspired by the principles of biological evolution. The key idea which governs these principles is the following: given a population of some *individuals* within an limited-resource environment, the competition between them in order to obtain more or better resources provokes a natural selection. Therefore, individuals winning the competition are the *best* individuals.

The same idea can be seccesfully adapted to an algorithm in order to solve an optimization problem. A set of *candidate solutions* is generated and a *fitness* value is attributed to each of them. Here, the analogy labels the candidate solutions as individuals, and individuals with a higher (or lower, with respect to the objective function of the problem) fitness value are the best, e.g., optimal, solutions. In the general idea, some of the best individuals are chosen as the new starting point for the next generation by applying mutation or recombination operators. The mutation operator is applied on a single individual and the result is a new individual, i.e., a candidate solution is mutated in a particular way in order to create a new candidate solution. The recombination operator is applied on two or more individuals called *parents* and it produces one or more individuals called *children*. When these two operators are applied on the parents the new set of generated individuals is called *offspring*. Each individual of this set is then valued w.r.t. the objective function, thus the fitness value is computed, and competes with the old individuals in order to be present in the next generation. This general process represents a loop and it is terminated by a particular stop criterion, which for instance can be a desired fitness value or a maximum number of evaluation.

It is useful to understand the flow of an evolutionary algorithm in order to approach how evolution strategy works. In Figure 4 a pseudocode is presented.

5.4.1 Components of an Evolutionary Algorithm

Before introducing the evolution strategy designed for the computation of the MPED, a better understanding of the various components of an evolutionary algorithm is needed. In

```
begin
    INITIALISE population $P$ with random candidate solutions;
    EVALUATE each candidate $p \in P$;
    repeat
        SELECT parents;
        RECOMBINE pairs of parents;
        MUTATE the resulting offsprings;
        EVALUATE new candidates;
        SELECT individuals for the next generation;
    until TERMINATION CRITERION is satisfied;
end
```

Algorithm 4: Pseudocode of an evolutionary algorithm.

this section we discuss the main components from the point of view of the application to the problem of computing the MPED.

Representation of individuals

As a first move in the direction of defining an evolutionary algorithm, we must abstract or simplify some aspects of the problem to create a well-defined context in which possible solutions can exist and can be evaluated. This work is often carried out by domain experts.

In the context of evolutionary algorithms, we refer to possible solutions within the original problem context as *phenotypes* while their encoding, that maps solutions to individuals, are called *genotypes*. The *representation* is the mapping from the phenotypes onto a set of genotypes which represent them. Obviously, there is no a particular representation for each possible problem context. For instance, given an optimization problem where the possible solutions are integers, phenotypes are surely formed by the given set of integers, whereas there could be different mappings forming the set of genotypes. As an example, the binary representation of the integer number could be used: in this case, the value 42 would be a phenotype, and 101010 a genotype representing it. At the end of the evolutionary algorithm, a solution would be a good phenotype and it can be obtained by decoding the best genotype.

We introduce the representation for our problem in Section 5.4.2.

Evaluation function

Evaluation function represents requisites that individuals would reach by adapting throughout the evolution. It constitutes the basis for the selection and indicates a criterion of improvement for an individual. We can see the evaluation function as a quality metric of an individual.

In the literature, the evaluation function is called *cost function* whether the optimization problem is a minimization problem, whereas it is called *fitness function* when the optimization problem is a maximization one. Also in this case, the terminology could be confusing; since an evolutionary algorithm usually solves an optimization problem, the term *objective function* might be used as well.

Population

Population is the set of (the representation of) possible solutions. Population is peculiarly important due to being the unit of evolution. In fact, individuals are static objects which do not change, while population intrinsically carries the concept of evolution and adaption. An important property of the population is the *population size*, that is how many individuals are in it. Usually, population size remains constant during time.

Parent Selection Mechanism

The parent selection mechanism allows to select the best individuals in order to become parents for the next generation. An individual is a *parent* if it has been selected to endure variations in order to create offspring.

Mutation

Mutation operator is applied on a single individual and it produces a modified individual. Usually, the mutation operator is always stochastic, i.e., introduced mutations are always due to a series of random choices. In general, mutation should be supposed to cause a random, unbiased change.

Thanks to the mutation operator it is possible to introduce new information within the population and consequently to explore the search space. It is straightforward to see how mutation operator allows to move in the space by generating new points, i.e., solutions. The magnitude of the mutation defines the distance between an individual and the one obtained by applying the mutation operator to it. The lesser the magnitude the closer the solutions in the space. In essence, the mutation operator (as many other variation operators) defines the evolutionary implementation of elementary (search) steps.

The kind of mutation operator depends on the problem, i.e., some mutation operators are better than others with respect to a particular representation. A specific methodology to define a mutation operator does not exist, albeit some rules have been proposed [15].

Crossover

The crossover operator combines information from two or more parents to one or more children. As the mutation operator, crossover is unbiased and stochastic. The main idea derives from the natural concept of hereditariness: joining two or more individuals with different but interesting characteristics or properties, a progeny which combines these characteristics might be generated.

Selection

In nature, selection is needed in order to distinguish individuals among their quality. In fact, selection operator leads the search in direction of the best promising zones including optimal solutions, thus giving a direction to the evolution. It is truly important to note that without selection, this evolutionary process would be the same as a random walking method. Individuals which will survive for the next generation are identified in the selection, based on their fitness level: it depends on the problem if either lower of higher is preferred. Selection operator is generally deterministic.

Initialization and Termination Criterion

Usually, the initialization of the individuals is random. However, some heuristics might be implemented in order to generate individuals with a good fitness value in the initialization.

When the problem has an optimal fitness value, ideally the termination criterion would represent the discovery of a solution with that exact fitness value. Similarly, the evolution process could stop when a solution is found with an optimal fitness value up to a difference of $\epsilon > 0$.

Most of the time, a guarantee to reach the optimal fitness value is not available. In order to to avoid a infinite execution, a termination criterion is needed. Some of the most common options could be, for instance, *(i)* to stop the execution after a maximum time T, *(ii)* to stop the execution after a certain number of calls to the fitness function or *(iii)* to continue the execution until the best fitness value found is greater (resp. lesser) than a certain threshold for a specific amount of time T.

5.4.2 Evolution Strategy

Evolution Strategy (ES from now on) belongs to the family of evolutionary algorithm. In the beginning, evolution strategies were not intended to compute the maximum or the minimum of some real-valued static function. Rather, they were used as a set of rules for the automatic design and analysis of consecutive experiments. First applications of ES were experimental; some examples dealt with drag minimization of a joint plate [98] and structure optimization of a two-phase flashing nozzle [101]. Due to the impossibility to describe and solve such problems in an analytical way, ES was simply used as an algorithm based on random changes of the experimental mechanisms. The idea was to apply random changes, to select the changes which performed better and to repeat the process until an optimal solution for the problem was found. We see that the first ES implementation effectively was in hardware, due to a lack of computational resources for high-precision simulations. Measurements of the objective function was obtained experimentally and changes, i.e., mutations, were introduced in the hardware.

ES can be seen as a specific instantiation of evolutionary algorithms and can be characterized by some specific properties [9], that are:

- Selection is a deterministic process;
- Mutation should be unbiased, i.e., it should not use any fitness information;
- Mutation operators are parameterized thus they can change their properties during optimization;
- Individuals consist of decision parameters as well as strategy parameters.

Depending on few parameters, different versions of ES exists and are briefly presented hereafter.

$(1+1)-ES$ variant

The first and simplest form of ES derived from the applications aforementioned. Since there was just one old individual (parent) and just one new individual (offspring) per iteration (generation) and since the selection took place among those two, this form of ES has been called $(1+1)-ES$ and referred as *two-membered*. This form represents the original version of ES, which was developed in order to approach discrete problems, it used mutations with a small magnitude and it was prone to stuck in a local optimum point. It has been proved that, given enough time, $(1+1)-ES$ reaches the global optimum of an objective function [103].

$(\mu+1)-ES$ variant

$(\mu+1)-ES$ represents the first generalization of $(1+1)-ES$. In this version, μ parents are maintained for each generation. Two of them are chosen at random and recombined to give life to an offspring, which also experiences mutation. The selection, thus, resembles the natural concept of "extinction of the worst" by cutting an offspring or one of the parents and keeping constant the population size.

$(\mu+\lambda)-ES$ and $(\mu,\lambda)-ES$ variants

Two further versions of multi-membered ES were introduced by Schwefel, which are:

- the $(\mu+\lambda)-ES$, in which $\lambda \geq 1$ descendants are created in a generation and the population size is mantained constant by discarding the λ worst individuals out of all $\mu+\lambda$ individuals;

- the $(\mu,\lambda)-ES$, where the selection is applied among the λ offspring only, in fact ignoring their parents and their fitness values. This strategy relies in a strict Darwinian sense of natural selection by having a birth surplus, i.e., $\lambda > \mu$. Essentially, parents do not survive for the successive generation, while μ individuals out of the λ offsprings are chosen as the parents for the successive generation.

In order to better understand the process which governs the ES, Algorithms 5 and 6 respectively show pseudocodes of the state-of-the-art $(1+1)-ES$ and $(\mu+\lambda)-ES$ algorithms capable of self-adaptation. To discuss all of the details within the algorithm is out of the scope for this thesis, while a discussion regarding the specialization of fundamental components of ES to the MPED is needed and presented in the following subsection.

5.4.3 Fundamental Components of Evolution Strategy for MPED

Individuals

The space of candidate solutions for the computation of MPED is represented by $\mathbb{M}$, i.e., the set of the possible matching schemas. Indeed, a solution of the problem is exactly a matching schema M, thus a representation in order to treat matching schemas as individuals is needed.

The representation used is the same as in hill climbing presented in Section 5.3.1. Simply, the order of the symbols is stored in an array, where each element $i \in \mathbb{N}_0$ indicates the i-th symbol in the lexicographically ordered juxtaposition of the symbols of the alphabet Π.

This representation results to be useful in order to apply mutation operators on a matching schemas.

Cost value

Cost value[1] for an individual M is given by the value of the edit distance computed using M as a matching schema, i.e., $\mathcal{L}^M(s_1, s_2)$.

Mutation operator

We designed and implemented different mutation operators which are briefly described next. Section 5.4.5 presents the result for each mutation operator.

Swap2 Randomly selects two elements of the permutation and swaps them.

Swap2-E A specialized version of Swap2 mutation operator for the MPED. It avoids to generate permutations which would result in giving the same value of MPED. Straightforwardly, whether $\pi_i = 1$, Swap2-E is the same as Swap2. As an example, Figure 5-11 illustrates a $\langle 3, 2\rangle$-matching schema. Suppose we could swap $a1$ and $a3$: the new obtained permutation would be equal to the original permutation, thus the mutation would produce an already existing individual.

[1]Since we are minimizing the edit distance, we refer to the cost value instead of fitness value.

Swap3 Mutates an individual by swapping three random elements between them. Positions of the element must be distinct.

Swap2-Swap3 A mutation operator which uses both Swap2 and Swap3 operators. Each time the operator is invoked, it randomly selects whether applying Swap2 or Swap3.

Scramble Selects a (contigue) part of the permutation and randomly repositions elements within it.

Inversion Selects a (contigue) part of the permutation and inverts the element in it.

Translocation Selects a (contigue) part of the permutation and moves it in another position without modifying the selected part.

5.4.4 Evolution Strategy Implementations

The strategies have been implemented using C++ language (C++11 standard). It is important to point out that crossover has not been used in the implementations, thus on generating offspring we cloned the selected parent.

Moreover, for the sake of simplicity, within the pseudocode context we made the following assumptions:

- a function *shuffle* exists which shuffles randomly a given integer representation of a matching schema;
- a function *mutate* exists and implements one of the designed mutation operators;
- an integer representation of a matching schema M possesses the attribute "costValue" which represents the MPED computed with M, i.e., $\mathcal{L}^M(s_1, s_2)$.

In the following, we briefly describe the ES variations which have been implemented. The experimental campaign will be discussed in Section 5.4.5.

$(1+1)-ES$

The implemented $(1+1)-ES$ variant exactly mimics the description given in Section 5.4.2. The termination condition has been parameterized: it is possible to stop the algorithm after a maximum number of generations or after a number of generations in which an improvement has not been obtained. Furthermore, it is possible to give a matching schema as a starting point, otherwise a random matching schema is selected.

Algorithm 5 shows the pseudocode of $(1+1)-ES$ implementation.

```
Input  : two strings $s_1$ and $s_2$ over the alphabets $\Pi_1$ and $\Pi_2$, respectively;
         a set $\chi$ of constraints;
         three integers $\pi_1$, $\pi_2$ and maxGeneration;
         two booleans StartingMatchingSchema and checkPlateau
Output: $\mathcal{L}_{\langle\pi_1,\pi_2,\chi\rangle}(s_1, s_2)$
Data   : plateau, currentGeneration: integers
         parent, child: matching schemas
begin
    parent = initialize();
    if startingMatchingSchema then
        parent = startingParent;
    else
        shuffle(parent);
        parent.costValue = $\mathcal{L}^{\text{parent}}(s_1, s_2)$;
    end
    plateau = 0;
    currentGeneration = 0;
    while currentGeneration < maxGeneration do
        child = parent;
        mutate(child);
        child.costValue = $\mathcal{L}^{\text{child}}(s_1, s_2)$;
        if child.costValue < parent.costValue then
            parent = child;
        else if checkPlateau then
            plateau = plateau + 1;
            if plateau == maxPlateau then
                return parent.costValue;
            end
        end
        currentGeneration = currentGeneration + 1;
    end
    return parent.costValue;
end
```

Algorithm 5: Pseudocode of $(1+1)-ES$.

$(1+1)-ES$ Restart

As opposed to the $(1+1)-ES$ variant, the restart property allows this variant to continue the execution, whether in presence of a plateau, by generating a random matching schema.

Using a feedback from the search context, it is possible to check if the obtained improvement in a range of generations is lower than a certain threshold and to decide whether to continue the research.

$(1+1)-ES$ Simple Restart

In this variant, the $(1+1)-ES$ Restart is executed a specified number k of times.

$(\mu+1)-ES$

A theoretical idea of this variant has been presented in Section 5.4.2. In this case, each generated offspring which experienced mutation is compared with the worst parent of the current generation. In this case, the number μ of parents to maintain at each generation has been parameterized.

$(\mu+\lambda)-ES$

A description of this variant has been given in Section 5.4.2. Also in this case, each component has been parameterized, thus it is possible to select the number μ of parents to maintain at and the number λ of offsprings to generate at each generation. Algorithm 6 shows the pseudocode of $(\mu+\lambda)$ implementation.

5.4.5 Experiments

A series of experiments to evaluate the reliability, precision and applicability of ES have been carried out. The main properties to determine and to analyze in this experimental campaign are: *(i)* precision of the ES and *(ii)* execution runtime.

The following part is organized as follows: in order to measure the precision, we started by comparing ES with an exhaustive approach. Due to the intractability of large instances, a general comparison with respect to Hill Climbing metaheuristic proposed in Section 5.3.1 has been carried out. Then, a series of tests in which a comparison between different mutation operators has been computed and analyzed. Finally, we compared all of the implemented ES variants.

For sake of simplicity, when a variant is not specified then $(\mu+1)-ES$ variant, together with the Swap2 operator, has been used.

```
Input  : two strings $s_1$ and $s_2$ over the alphabets $\Pi_1$ and $\Pi_2$, respectively;
         a set $\chi$ of constraints;
         five integers $\pi_1$, $\pi_2$, maxGeneration, $\mu$ and $\lambda$;
         one boolean StartingMatchingSchema
Output: $\mathcal{L}_{\langle\pi_1,\pi_2,\chi\rangle}(s_1, s_2)$
Data   : currentGeneration: integers
         poolParents, poolChildren: set of matching schemas
begin
    poolParents = $\emptyset$ if startingMatchingSchemas then
        poolParents = startingPoolParents;
    else
        for $i = 0 \to \mu$ do
            parent = generateRandomParent();
            poolParents.insert(parent);
        end
    end
    currentGeneration = 0;
    while currentGeneration < maxGeneration do
        poolChildren = $\emptyset$;
        for $j = 0 \to \lambda$ do
            child = selectRandomParent(poolParents);
            mutate(child);
            child.costValue = $\mathcal{L}^{\text{child}}(s_1, s_2)$;
            poolChildren.insert(child);
        end
        poolParents = selectBestIndividuals(poolChildren, poolParents);
        currentGeneration = currentGeneration + 1;
    end
    bestParent = selectBestIndividual(poolParents);
    return bestParent.costValue;
end
```

Algorithm 6: Pseudocode of $(\mu + \lambda)-ES$.

Precision (vs Exhaustive Approach)

Tests Configuration To carry out this test, three different configurations of $(\mu+\lambda)-ES$, namely *strong*, *medium* and *light*, have been used. Configurations differ by the maximum number of generations. In particular, Table 5.20 shows used values for the respective parameters.

Dataset Due to the same nature of the experiment, we used the same datasets generated for the analysis of reliability of HC and SA discussed in Section 5.3.3.

Configuration	μ	λ	N. of generations
strong	30	120	120
medium	30	120	60
light	30	120	30

Table 5.20: Parameters and values for each configuration of $(\mu + \lambda) - ES$ vs an exhaustive approach.

Results In order to achieve coherent statistical results using ES heuristic, each test has been carried out 10 times and the average value has been used as solution. Precision is computed as follows: let d_{EX} be the solution of an instance using the exhaustive approach, i.e., the optimum, and let d_{ES} be the ES solution. We define precision P_{ES} as

$$P_{ES} = 1 - \frac{d_{ES} - d_{EX}}{d_{EX}} \tag{5.2}$$

Table 5.21, 5.22 and 5.23 respectively report results for each configuration, i.e., *strong*, *medium* and *light*. Note that when $P_{ES} = 1.00$, ES reaches the same solution as the exhaustive approach. A cell reporting N/A indicate an instance for which the exhaustive approach could not be computed.

It is important to point out that, for each configuration, ES successfully reaches a precision equal or very close to 1.00. These results shows how the evolutionary mechanism, strongly based on a stochastic process, becomes useful to effectively explore a large search space.

		len(*s*)				
		50	100	200	350	500
π	$\|\Pi\|$	$P_{(\mu+\lambda)-ES}$	$P_{(\mu+\lambda)-ES}$	$P_{(\mu+\lambda)-ES}$	$P_{(\mu+\lambda)-ES}$	$P_{(\mu+\lambda)-ES}$
	3	1.00	1.00	1.00	1.00	1.00
	4	1.00	1.00	1.00	1.00	1.00
	5	1.00	1.00	1.00	1.00	1.00
	6	1.00	1.00	1.00	1.00	1.00
1	7	1.00	1.00	1.00	1.00	0.98
	8	1.00	1.00	1.00	1.00	1.00
	9	1.00	1.00	1.00	0.99	1.00
	10	1.00	0.98	1.00	1.00	1.00
	5	1.00	1.00	1.00	1.00	1.00
2	6	1.00	1.00	1.00	1.00	1.00
	7	1.00	1.00	1.00	*N/A*	*N/A*
3	7	1.00	1.00	0.98	*N/A*	*N/A*

Table 5.21: Results for $(\mu+\lambda)-ES$ with configuration *strong*.

		len(*s*)				
		50	100	200	350	500
π	$\|\Pi\|$	$P_{(\mu+\lambda)-ES}$	$P_{(\mu+\lambda)-ES}$	$P_{(\mu+\lambda)-ES}$	$P_{(\mu+\lambda)-ES}$	$P_{(\mu+\lambda)-ES}$
	3	1.00	1.00	1.00	1.00	1.00
	4	1.00	1.00	1.00	1.00	1.00
	5	1.00	1.00	1.00	1.00	1.00
	6	1.00	1.00	1.00	1.00	1.00
1	7	1.00	1.00	1.00	1.00	1.00
	8	1.00	1.00	0.98	1.00	1.00
	9	1.00	1.00	1.00	1.00	0.00
	10	1.00	0.98	0.99	1.00	1.00
	5	1.00	1.00	1.00	1.00	1.00
2	6	1.00	1.00	1.00	1.00	1.00
	7	1.00	1.00	1.00	*N/A*	*N/A*
3	7	1.00	1.00	0.98	*N/A*	*N/A*

Table 5.22: Results for $(\mu+\lambda)-ES$ with configuration *medium*.

		$len(s)$				
		50	100	200	350	500
π	$\|\Pi\|$	$P_{(\mu+\lambda)-ES}$	$P_{(\mu+\lambda)-ES}$	$P_{(\mu+\lambda)-ES}$	$P_{(\mu+\lambda)-ES}$	$P_{(\mu+\lambda)-ES}$
	3	1.00	1.00	1.00	1.00	1.00
	4	1.00	1.00	1.00	1.00	1.00
	5	1.00	1.00	1.00	1.00	1.00
	6	1.00	1.00	1.00	1.00	1.00
1	7	1.00	1.00	1.00	1.00	1.00
	8	1.00	0.98	0.97	1.00	1.00
	9	1.00	1.00	0.99	1.00	1.00
	10	1.00	1.00	100	1.00	1.00
	5	1.00	1.00	1.00	1.00	1.00
2	6	1.00	1.00	1.00	1.00	1.00
	7	1.00	1.00	0.99	N/A	N/A
3	7	1.00	1.00	1.00	N/A	N/A

Table 5.23: Results for $(\mu + \lambda)-ES$ with configuration *light*.

Parameter	Value
μ	30
λ	120
Maximum n. of generations	120

Table 5.24: Selected values for ES parameters.

Runtime (vs Hill Climbing)

Tests Configuration For this configuration, $(\mu + \lambda) - ES$ variant has been chosen as candidate. The test carried out determines how the runtime of ES compares to HC, which we presented in Section 5.3.1. Furthermore, in addition to the runtime, the obtained solution is accounted too. Table 5.24 shows selected values for the respective parameters.

Dataset As for the precision experiment, a set of instances have been created, using the following ranges of values:

- $len(s) = \{1000, 2000, 3500\}$,
- $|\Pi| = \{14, 16, 18, 20\}$,
- $\pi_i = \{1..10\}$ and such that $\pi_i < \lceil |\Pi|/2 \rceil$.

Results (graphs) We decided to show results for a subset of parameters values and for a clearer presentation few graphs have been produced. Graphs have been divided in two parts:

- Figures 5-15, 5-16, 5-17 and 5-18 report results for different values of the parameter $|\Pi|$, having others as fixed;
- Figures 5-19, 5-20, 5-21 and 5-22 instead report results for different values of $\pi_1 = \pi_2$, also having others parameters as fixed.

After analyzing the results, we can notice how, for both the runtime and the optimality of the obtained solutions, ES performs significantly better than HC. In fact, ES reaches a better solution in a significantly lower number of iterations[2]. Also, from the obtained solution we extrapolated the related matching schema M and we correctly assured that M was a valid matching schema for the instance.

[2]We will also show in a few experiments that the running time is also significantly smaller.

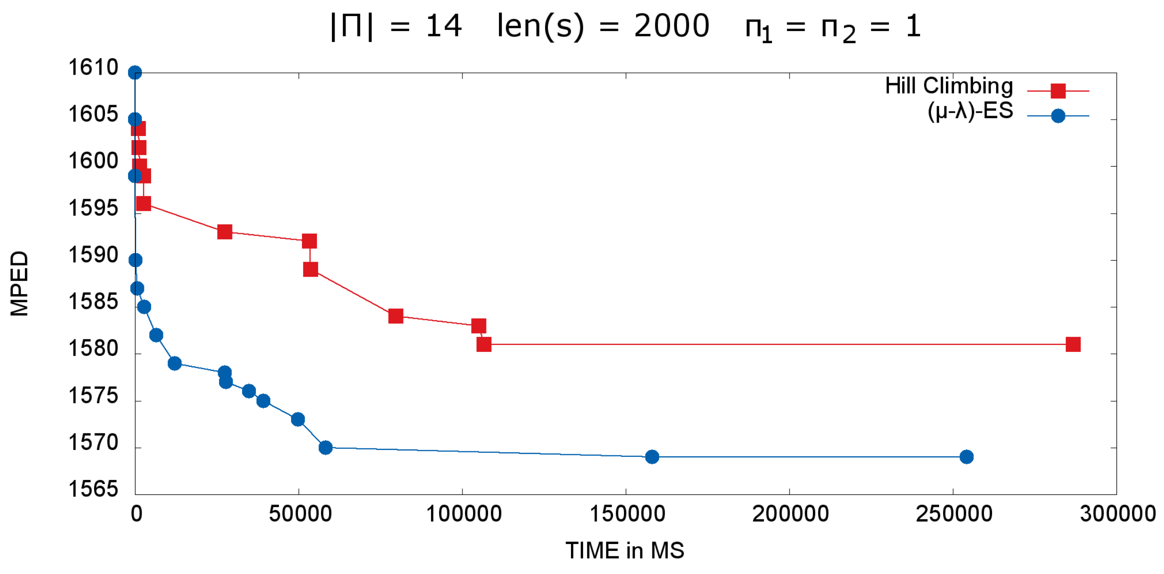


Figure 5-15: Runtime comparison of ES and HC with $|\Pi| = 14, len(s) = 2000, \pi_1 = \pi_2 = 1$

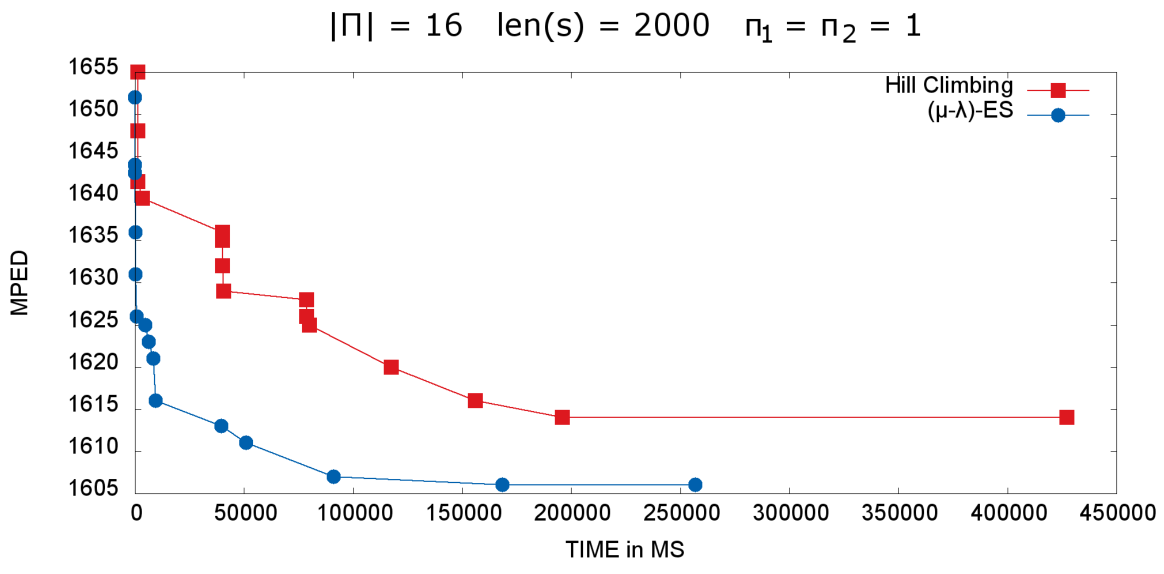


Figure 5-16: Runtime comparison of ES and HC with $|\Pi| = 16, len(s) = 2000, \pi_1 = \pi_2 = 1$

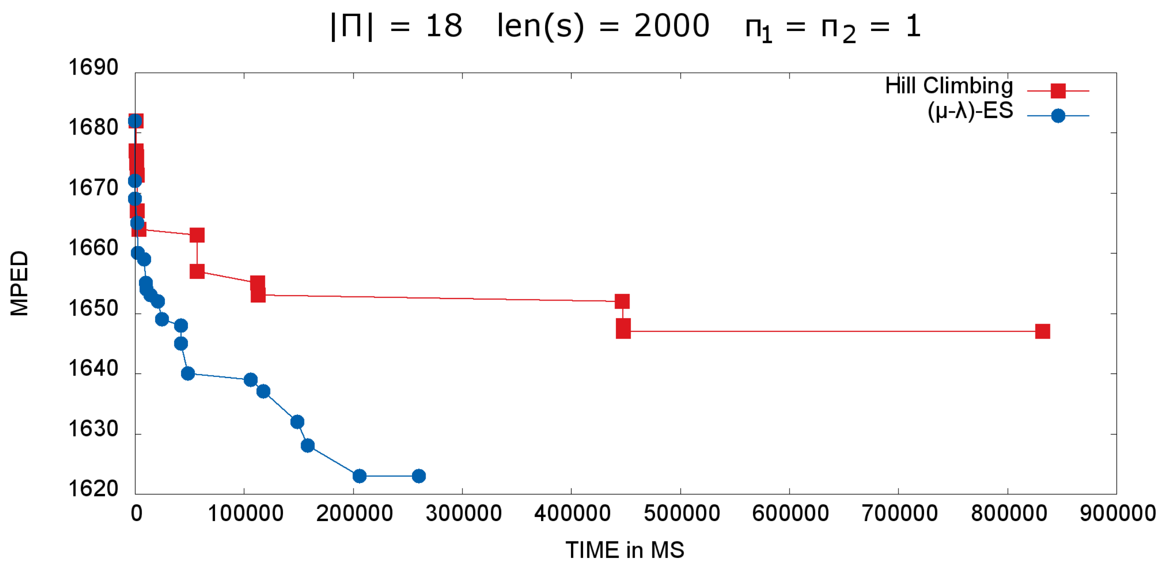


Figure 5-17: Runtime comparison of ES and HC with $|\Pi| = 18, len(s) = 2000, \pi_1 = \pi_2 = 1$

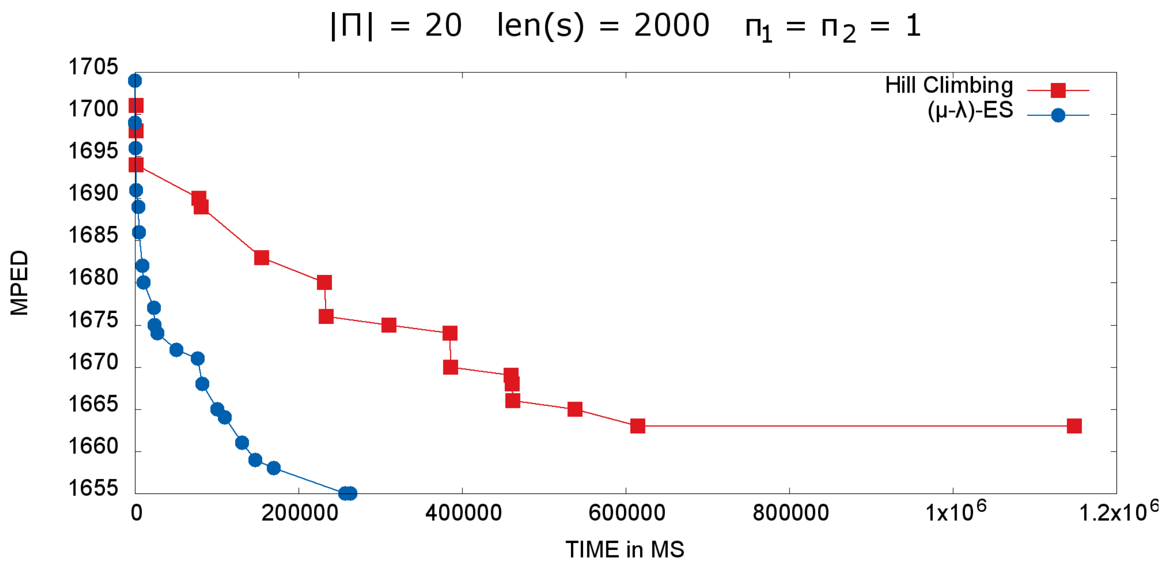


Figure 5-18: Runtime comparison of ES and HC with $|\Pi| = 20, len(s) = 2000, \pi_1 = \pi_2 = 1$

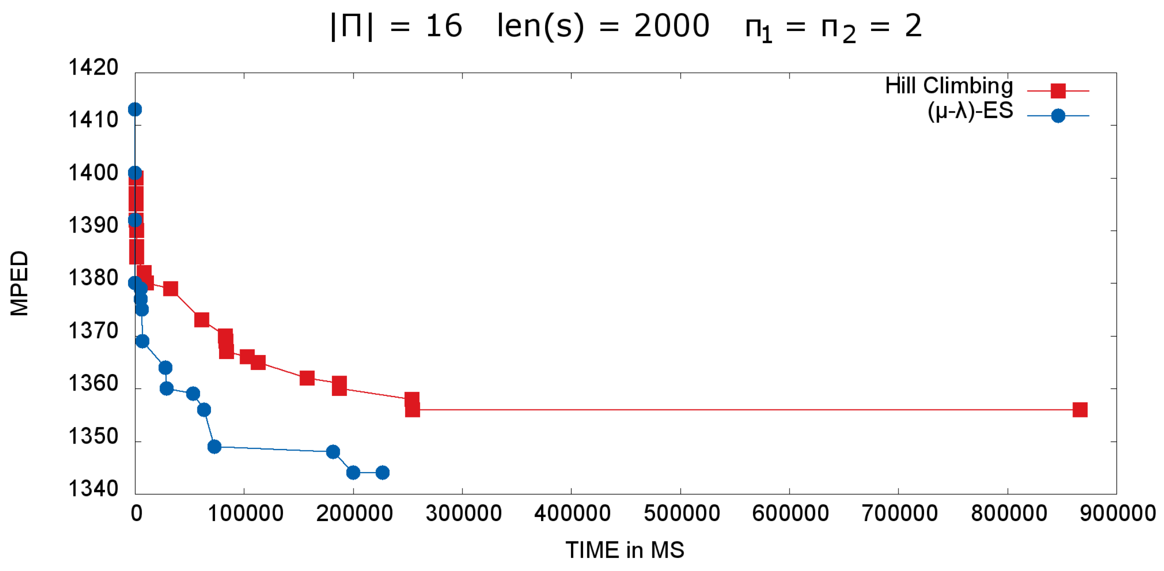


Figure 5-19: Runtime comparison of ES and HC with $|\Pi| = 16, len(s) = 2000, \pi_1 = \pi_2 = 2$

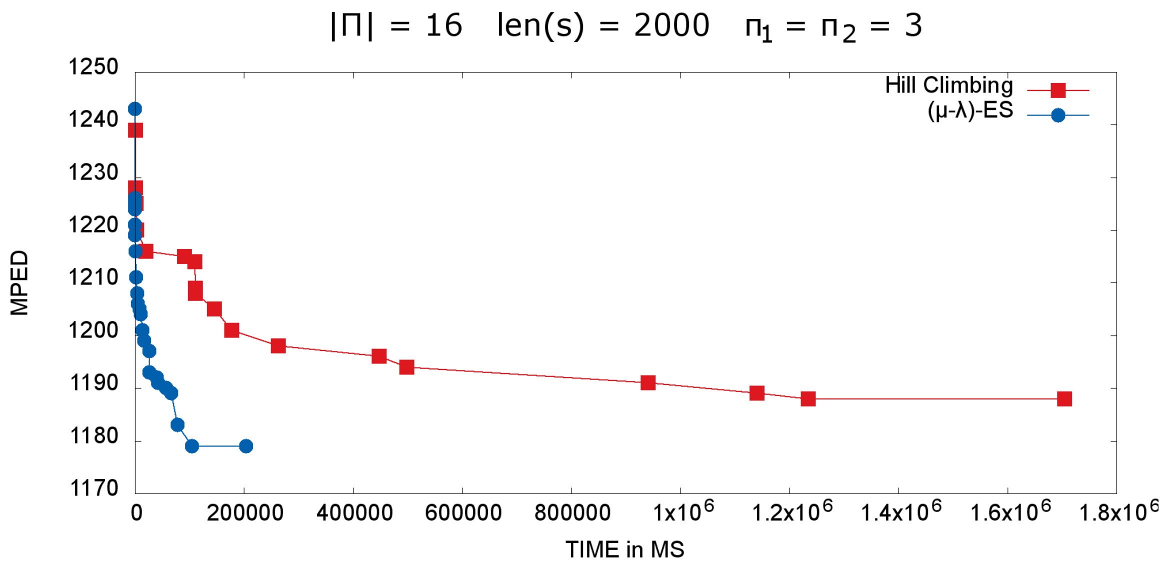


Figure 5-20: Runtime comparison of ES and HC with $|\Pi| = 16, len(s) = 2000, \pi_1 = \pi_2 = 3$

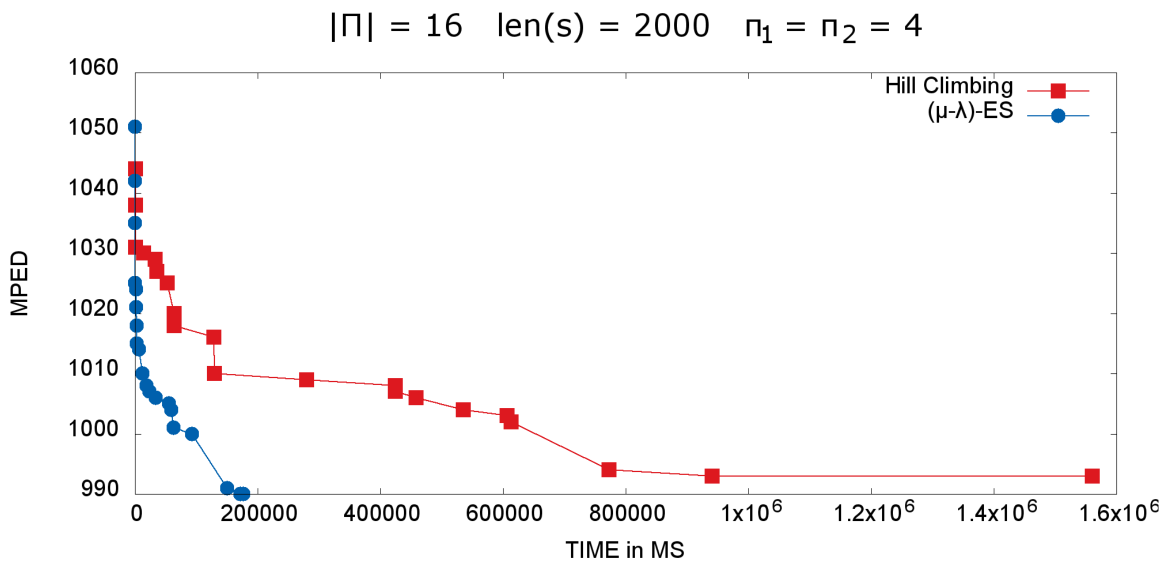


Figure 5-21: Runtime comparison of ES and HC with $|\Pi| = 16, len(s) = 2000, \pi_1 = \pi_2 = 4$

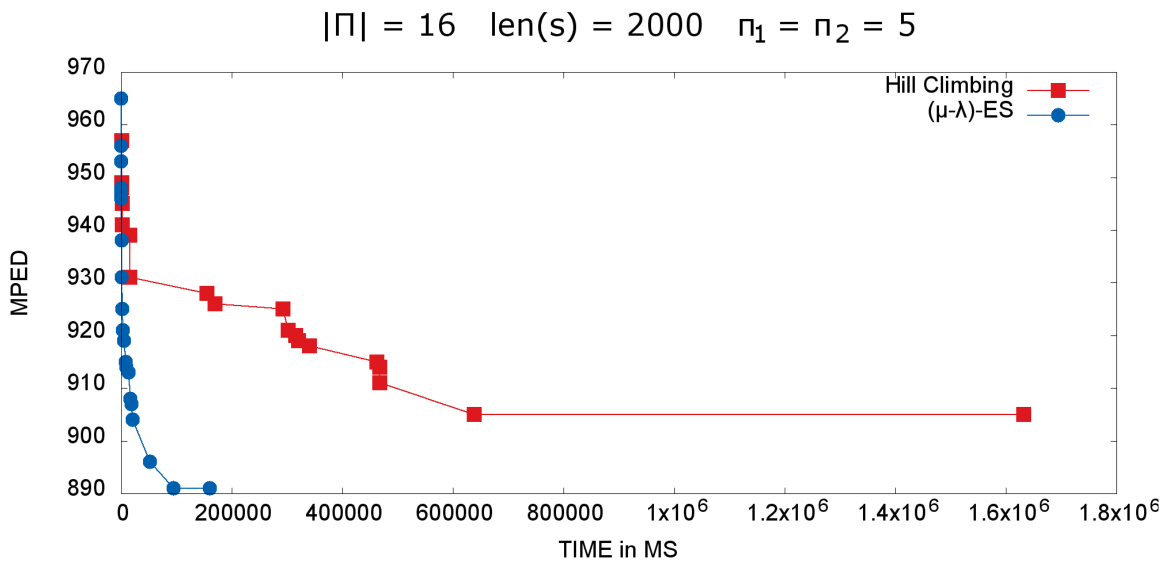


Figure 5-22: Runtime comparison of ES and HC with $|\Pi| = 16, len(s) = 2000, \pi_1 = \pi_2 = 5$

Results (tables) To account all of the results, a tabular view of them is presented. Moreover, an analyis of precision has been carried out too. Due to being unfeasible to obtain exact solution for these problem instances, we estimated the precision of ES w.r.t the ad-hoc HC implementation, defining it as

$$P_{ES} = 1 - \frac{d_{ES} - min(d_{ES}, d_{HC})}{min(d_{ES}, d_{HC})} \tag{5.3}$$

where d_{HC} is the solution obtained by HC. It is useful to underline that $P_{ES} = 1.00$ implies that ES, for that specific instance, has a higher than or at least equal to HC. Also in this case, for a statistic coherence, both ES and HC have been executed 10 times for each instance and the average values has been used as solution.

Table 5.25 summarizes the precision comparison. Table 5.26 compares ES runtime T_{ES} with HC runtime T_{HC}.

Observing the results, we conclude that ES performs significantly better than HC, both from the precision and runtime points of view. on average, ES results 13.6 times faster than HC and only in few cases, runtime is slightly similar. In the best case ($|\Pi| = 20$, $len(s) = 1000$, $\pi_i = 6$), ES is 100 times faster than HC and maintains an estimated precision of 0.99. In the worst case ($|\Pi| = 14$, $len(s) = 3500$, $\pi_i = 1$), ES' runtime is close to HC's but the runtime ratio is 0.97.

		$len(s)$		
		1000	2000	3500
π	$\|\Pi\|$	$P_{(\mu+\lambda)\text{–}ES}$	$P_{(\mu+\lambda)\text{–}ES}$	$P_{(\mu+\lambda)\text{–}ES}$
1	14	1.00	1.00	1.00
	16	1.00	1.00	1.00
	18	1.00	1.00	1.00
	20	1.00	1.00	1.00
2	14	1.00	1.00	1.00
	16	1.00	1.00	1.00
	18	1.00	1.00	1.00
	20	1.00	1.00	1.00
3	14	1.00	1.00	1.00
	16	1.00	1.00	1.00
	18	1.00	1.00	1.00
	20	1.00	1.00	1.00
4	14	1.00	1.00	1.00
	16	1.00	1.00	1.00
	18	1.00	1.00	1.00
	20	0.99	1.00	1.00
5	14	1.00	1.00	1.00
	16	1.00	1.00	1.00
	18	1.00	1.00	1.00
	20	1.00	1.00	1.00
6	14	1.00	1.00	1.00
	16	1.00	1.00	1.00
	18	1.00	1.00	1.00
	20	0.99	1.00	1.00
7	14	1.00	1.00	1.00
	16	1.00	1.00	1.00
	18	0.99	1.00	1.00
	20	0.99	1.00	1.00
8	16	1.00	1.00	1.00
	18	1.00	1.00	1.00
	20	1.00	1.00	1.00
9	18	1.00	0.99	1.00
	20	1.00	1.00	1.00
10	20	1.00	1.00	1.00

Table 5.25: Obtained precision of $(\mu+\lambda)-ES$ w.r.t HC.

		len(s)		
		1000	2000	3500
π	$\|\Pi\|$	$T_{(\mu+\lambda)-ES}/T_{HC}$	$T_{(\mu+\lambda)-ES}/T_{HC}$	$T_{(\mu+\lambda)-ES}/T_{HC}$
	14	0.90	0.89	0.97
1	16	0.45	0.60	0.74
	18	0.42	0.31	0.21
	20	0.23	0.23	0.15
	14	0.29	0.36	0.53
2	16	0.30	0.26	0.25
	18	0.10	0.12	0.15
	20	0.07	0.11	0.07
	14	0.12	0.14	0.44
3	16	0.18	0.12	0.10
	18	0.11	0.10	0.08
	20	0.03	0.03	0.02
	14	0.26	0.14	0.12
4	16	0.06	0.11	0.11
	18	0.09	0.06	0.10
	20	0.06	0.03	0.04
	14	0.09	0.18	0.11
5	16	0.04	0.10	0.08
	18	0.03	0.07	0.08
	20	0.04	0.03	0.03
	14	0.09	0.15	0.16
6	16	0.07	0.08	0.13
	18	0.06	0.08	0.05
	20	0.01	0.03	0.05
	14	0.29	0.15	0.24
7	16	0.10	0.06	0.07
	18	0.03	0.05	0.06
	20	0.02	0.03	0.05
	16	0.08	0.09	0.07
8	18	0.05	0.06	0.04
	20	0.02	0.02	0.06
9	18	0.08	0.05	0.03
	20	0.03	0.02	0.04
10	20	0.04	0.05	0.06

Table 5.26: Runtime analysis of $(\mu+\lambda)-ES$ w.r.t HC.

Mutation Operators Tests

Tests Configuration Mutation operators have been tested using the $(\mu+1)-ES$ variant. Table 5.27 indicates selected values for the parameters. For each mutation operator, the variant has been executed 100 times.

Parameter	Value
μ	5
Maximum n. of generations	3600

Table 5.27: Selected values for ES parameters within mutation operators tests.

Dataset Also for this test, a dataset with the same conditions as of previous experiments has been generated, using the following ranges of values:

- $len(s) = 1000$,
- $|\Pi| = 20$,
- $\pi_i = \{1..10\}$.

Results (graphs) To visualize the results for each mutation operators and to compare them at the same time, box-and-whisker diagrams have been produced. By analyzing the obtained results, we first note how Swap2, Swap2-E and Swap2-Swap3 represent the mutation operators which the lowest MPED and the lowest median MPED. When π_i increases, the behaviour of Swap3 seems to mimic that of the aforementioned mutation operators: the intuition is that for increasing values of π_i, Swap3 modifies different π_i-partitions, thus effectively perturbs the matching schema. Also, Swap2 shows the smaller interquartile range due to the specialization of avoiding to generate permutations which would result in the same value of MPED. We conclude that mutation operators which manipulate (large) portions of the array used to represent the matching schemas are less effective than the ones which swap symbols, due to the fact that the formers do not introduce an effective variation.

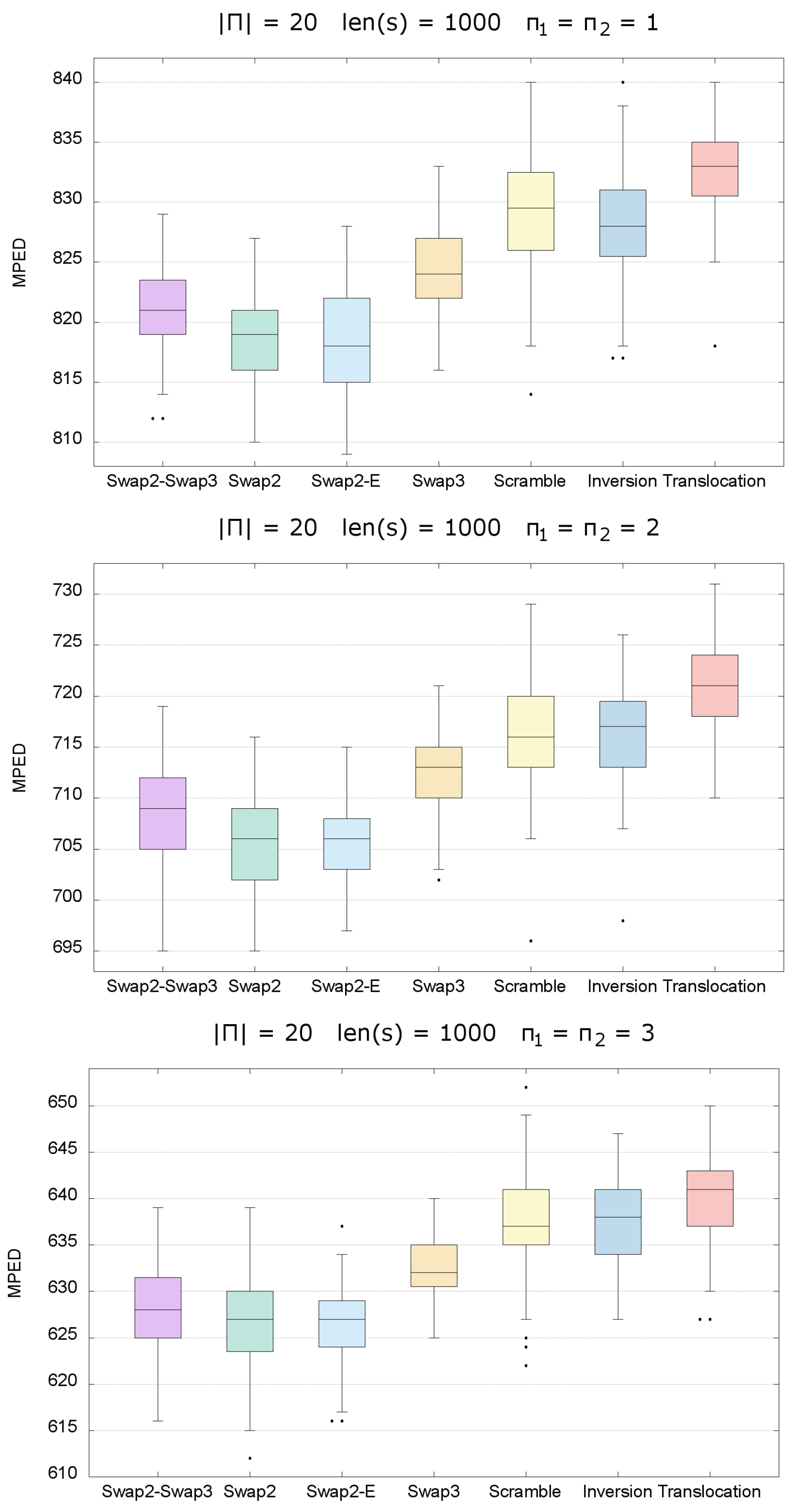


Figure 5-23: Comparison of mutation operators with $|\Pi| = 20, len(s) = 1000, \pi_1 = \pi_2 = \{1, 2, 3\}$

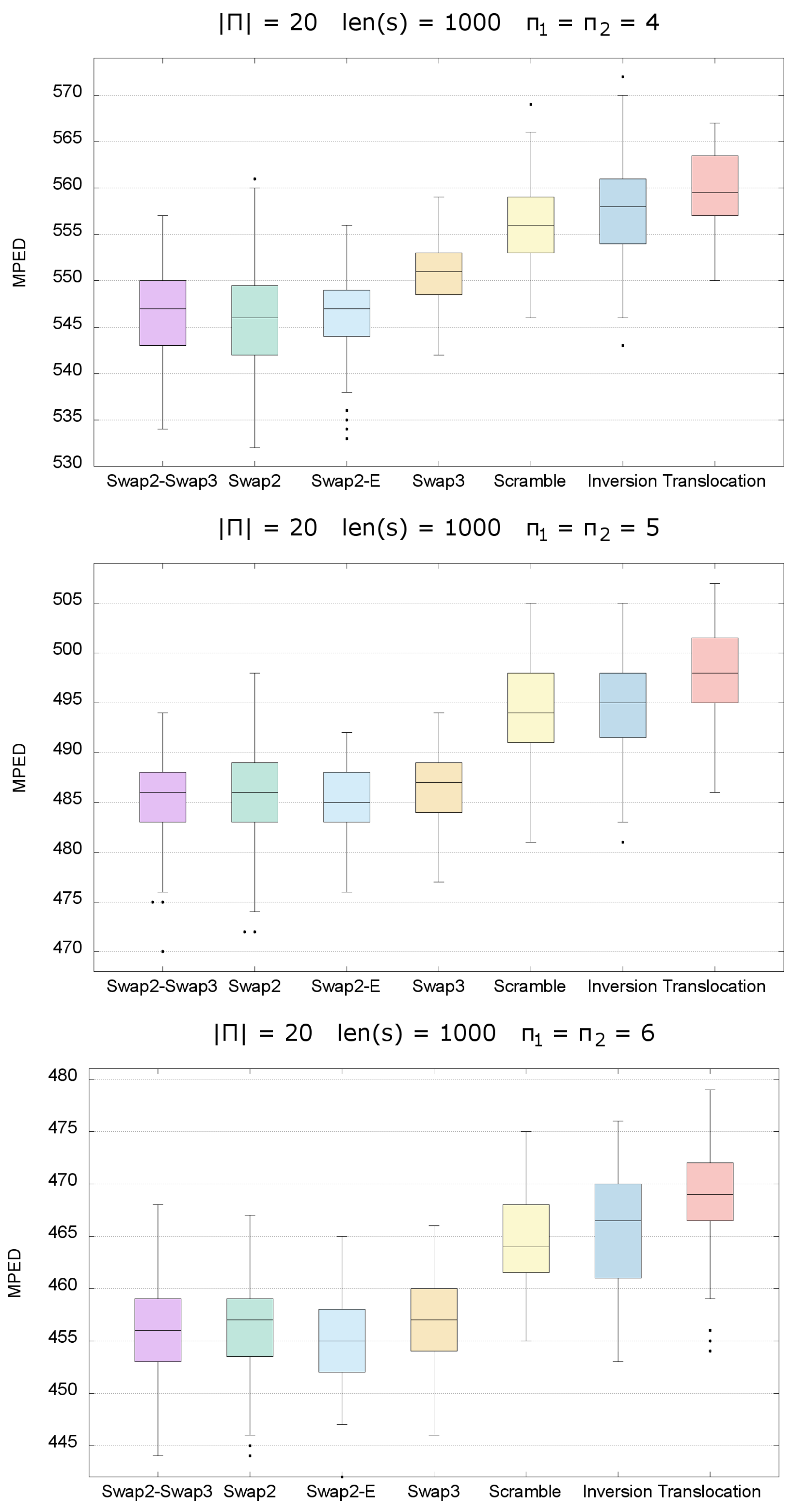


Figure 5-24: Comparison of mutation operators with $|\Pi| = 20, len(s) = 1000, \pi_1 = \pi_2 = \{4, 5, 6\}$

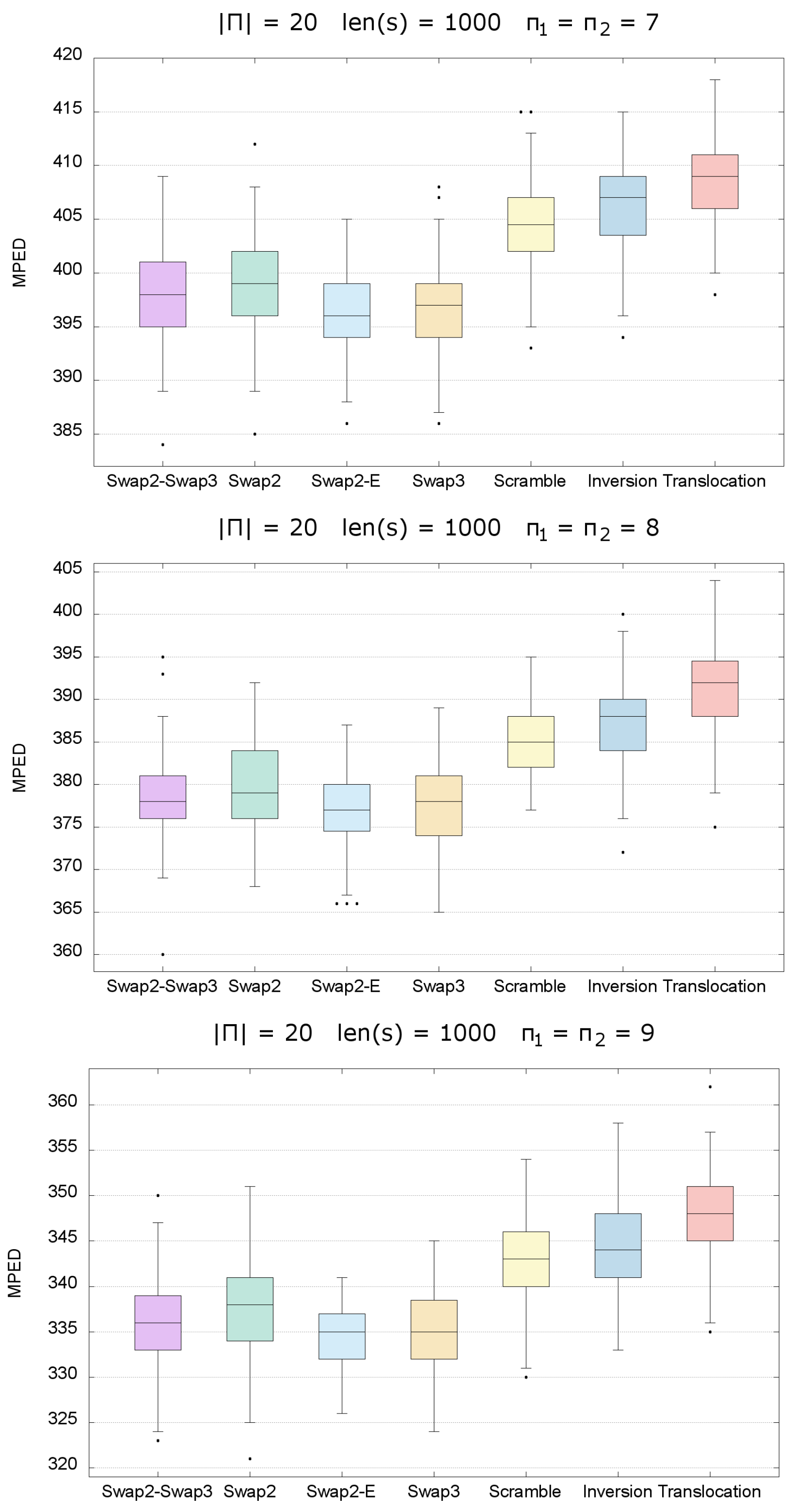


Figure 5-25: Comparison of mutation operators with $|\Pi| = 20, len(s) = 1000, \pi_1 = \pi_2 = \{7, 8, 9\}$

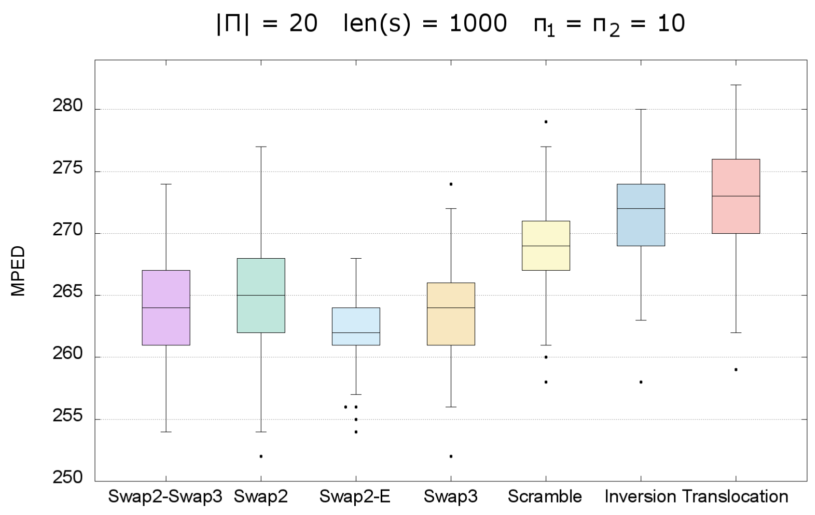


Figure 5-26: Comparison of mutation operators with $|\Pi| = 20, len(s) = 1000, \pi_1 = \pi_2 = 10$

Results (tables) We estimated the precision P_M defined as

$$P_M = 1 - \frac{d_M - min(\Theta^i)}{min(\Theta^i)} \tag{5.4}$$

for each mutation operator used, where d_M is the median value of 100 executions for a specific mutation operator and Θ^i is the set of median values of all of the mutation operators tested for $\pi = i, i \in \{1..10\}$. Note that d_M represents the obtained MPED.
Table 5.28 shows obtained results. A cell containing 1.00 indicates that the corresponding mutation operator obtained the lowest median value for the corresponding value of π_i. Here, we distinguish the same trend observed in the graphical results. Precision of mutation operators which manipulate (large) portions of the permutations obtain the lowest precision, with the minimum equals to 0.96, obtained by the Translocation mutation operator. It is important to note how Swap2-E, which represents a specialized version of Swap2 for MPED, obtained a precision of 1.00 for each value of π_i.

Precision among ES variants

There are no significative differences in runtimes of different ES variants. As a consequence, we next concentrate on the precision analysis for those variants that best performed in previous tests. In particular, we considered:

- $(1+1)-ES$, allowing at most 14400 generations,

π	Swap2-E	Swap2	Swap3	Swap2-Swap3	Inversion	Scramble	Translocation
1	1.00	1.00	0.99	1.00	0.99	0.99	0.98
2	1.00	1.00	0.99	1.00	0.98	0.99	0.98
3	1.00	1.00	0.99	1.00	0.98	0.98	0.98
4	1.00	1.00	0.99	1.00	0.98	0.98	0.98
5	1.00	1.00	1.00	1.00	0.98	0.98	0.97
6	1.00	1.00	1.00	1.00	0.97	0.98	0.97
7	1.00	0.99	1.00	0.99	0.97	0.98	0.97
8	1.00	0.99	1.00	1.00	0.97	0.98	0.96
9	1.00	0.99	1.00	1.00	0.97	0.98	0.96
10	1.00	0.99	0.99	0.99	0.96	0.97	0.96

Table 5.28: Estimated precision P_M for each mutation operator for different values of π_i.

- $(1+1)-ES$ Simple Restart, allowing 10 attempts and at most 1440 generations per attempt,
- $(\mu+1)-ES$, with $\mu = 30$ and allowing at most 14400 generations,
- $(\mu+\lambda)-ES$ with $\mu = 30, \lambda = 120$ and allowing at most 120 generations.

Each variants used Swap2-E mutation operator and has been executed 100 times for each problem instance.

Dataset As we have done for the other tests, a set of problem instances has been generated, using the following ranges of values:

- $len(s) = 1000$,
- $|\Pi| = 20$,
- $\pi_i = \{1..10\}$.

Results (graphs) In the following, a series of box-and-whisker diagrams have been produced, reporting values of MPED obtained by the used variants. From the graphs, we see that $(\mu+\lambda)-ES$ and $(\mu+1)-ES$ are the best variants in this test, albeit the lowest MPED is often reached also by $(1+1)-ES$ variant. The Simple Restart variant does not perform as the others, probably due to having a lower number of generations per attempt; in fact, a restart might drive the heuristic in a point in the search space close to the one in which it previously terminated. In conclusion, $(\mu+\lambda)-ES$ and $(\mu+1)-ES$ variants perform better and with a smaller interquartile range than $(1+1)-ES$ variant.

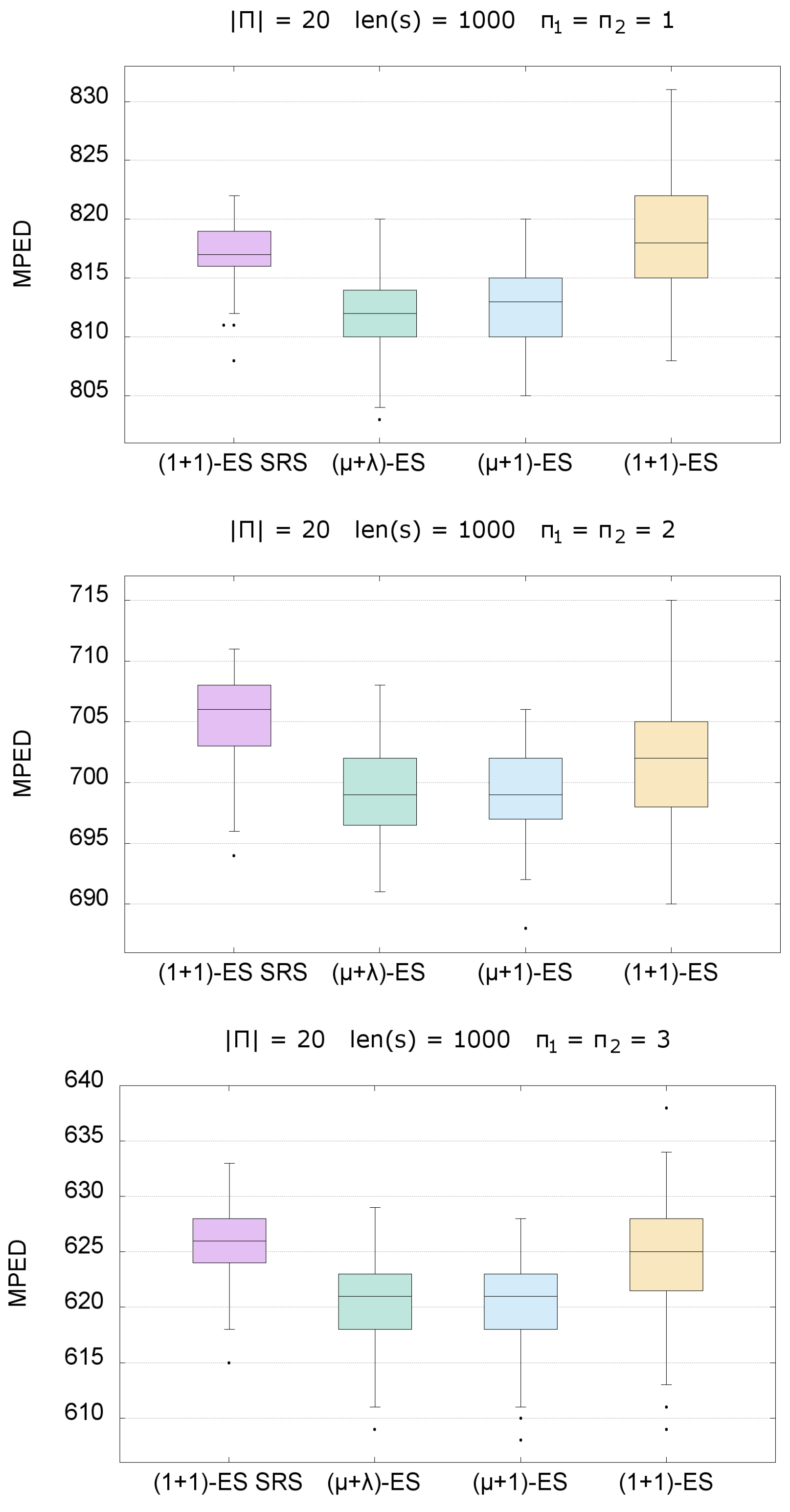


Figure 5-27: Comparison of ES variants with $|\Pi| = 20, len(s) = 1000, \pi_1 = \pi_2 = \{1, 2, 3\}$

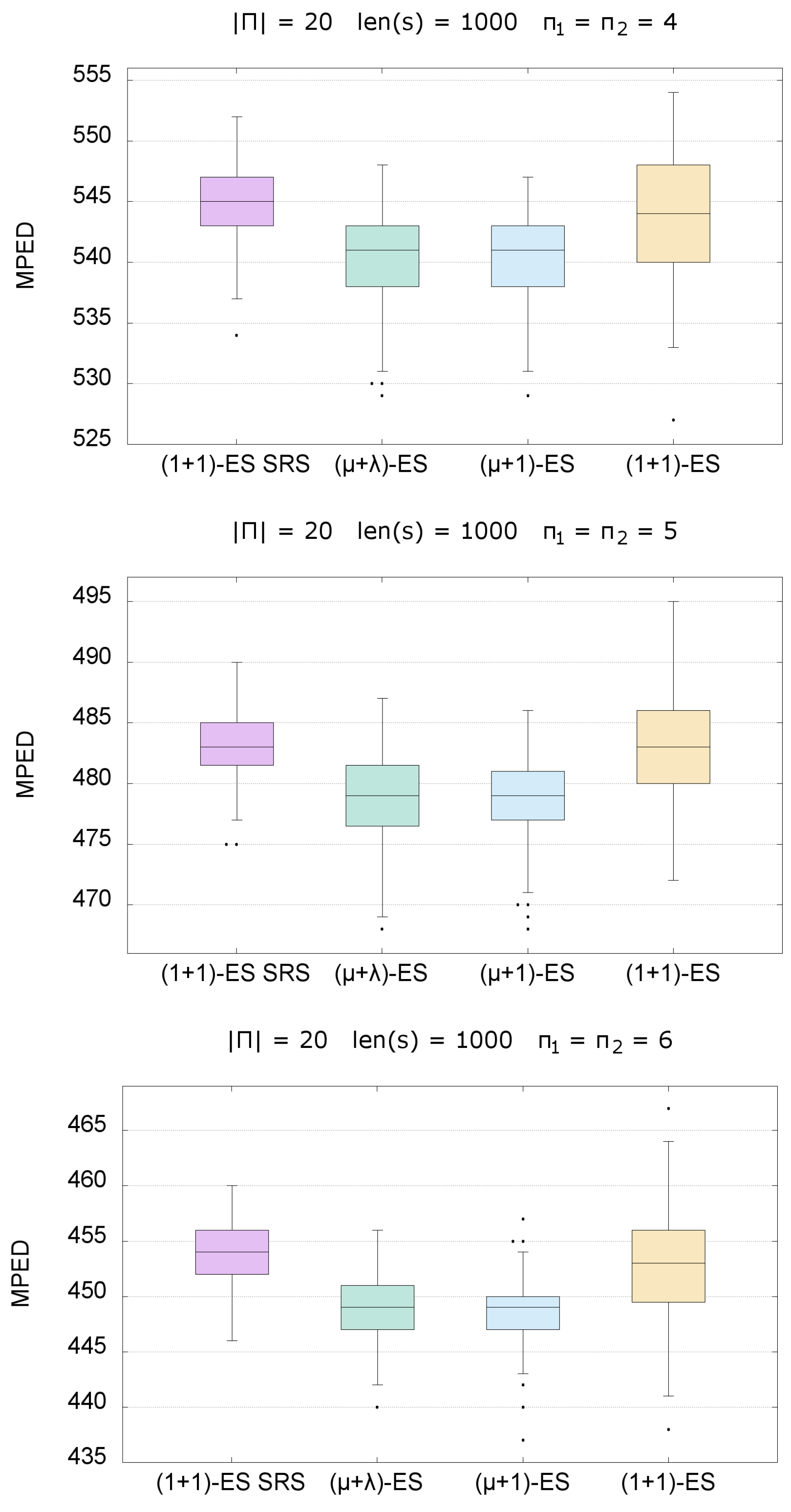


Figure 5-28: Comparison of ES variants with $|\Pi| = 20, len(s) = 1000, \pi_1 = \pi_2 = \{4, 5, 6\}$

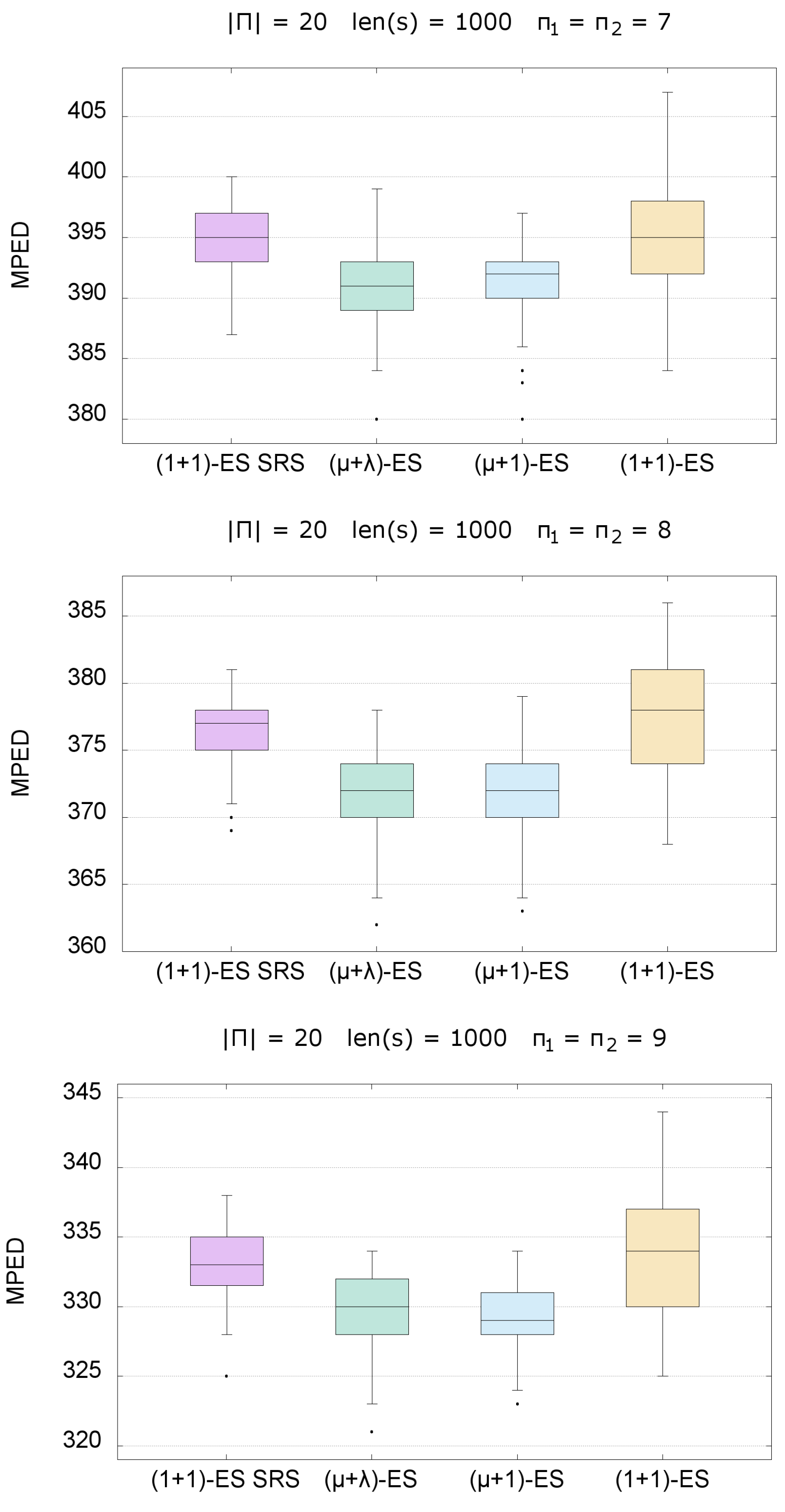


Figure 5-29: Comparison of ES variants with $|\Pi| = 20, len(s) = 1000, \pi_1 = \pi_2 = \{7, 8, 9\}$

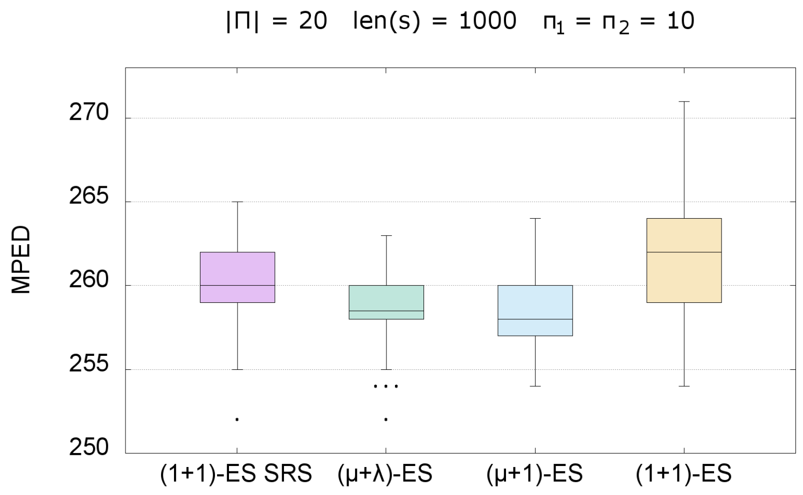


Figure 5-30: Comparison of ES variants with $|\Pi| = 20, len(s) = 1000, \pi_1 = \pi_2 = 10$

Results – Table Similarly to what we have done in previous tests, we estimated the precision P_V as

$$P_V = 1 - \frac{d_V - min(\Theta^i)}{min(\Theta^i)} \tag{5.5}$$

for each selected ES variant, where d_V is the median value of 100 executions for a specific variant and Θ^i is the set of median values of all of the variants tested for $\pi = i, i \in \{1..10\}$. Also in this case, d_V represents the obtained MPED by a variant on a specific instance. Table 5.29 shows obtained results. A cell containing 1.00 indicates that the corresponding variant obtained the lowest median value for the corresponding value of π_i.

π	$(\mu+\lambda)$-ES	$(\mu+1)$-ES	(1+1)-ES SRS	(1+1)-ES
1	1.00	1.00	0.99	0.99
2	1.00	1.00	0.99	1.00
3	1.00	1.00	0.99	0.99
4	1.00	1.00	0.99	0.99
5	1.00	1.00	0.99	0.99
6	1.00	1.00	0.99	0.99
7	1.00	1.00	0.99	0.99
8	1.00	1.00	0.99	0.98
9	1.00	1.00	0.99	0.98
10	1.00	1.00	0.99	0.98

Table 5.29: Estimated precision P_V for each ES variant and different values of π.

As results show, there are no distinguished differences between ES variants. In fact, the most significant difference, albeit irrelevant, is found on the instance with $\pi_i = 8$, where $(1+1)-ES$ returns a median value of 378 whereas $(\mu+\lambda)-ES$ returns 372.

Probability Distribution Tests

One last interesting analysis is the quantification of the quality of ES heuristic, i.e., the guarantee (or the probability to guarantee) that ES reaches a *good* solution within a certain time interval. The next tests measure the probability that ES reaches a certain level of accuracy within a certain amount of time.

For the following tests, $(\mu + \lambda) - ES$ has been used, equipped with Swap2-E mutation operator. For a better comprehension of the behaviour of the variant, two different configurations have been used: Table 5.30 shows both configurations with respective parameters.

N. of evaluations	μ	λ	Maximum n. of generations
3600	30	120	30
14400	30	120	120

Table 5.30: Configurations of $(\mu + \lambda) - ES$ for probability distribution test.

Dataset A set of problem instances has been generated, along the same lines as the previous tests, using the following ranges of values:

- $len(s) = 1000$,
- $|\Pi| = 20$,
- $\pi_i = \{1..10\}$.

Both configurations have been executed 100 times for each problem instance.

Results (graphs) Given X as the minimum value among the set of 100 runs, the following graphs show the probability of obtaining a value of MPED $x \in [X, X + \Delta]$, where Δ is an increment of the obtained MPED which depends on the number of evaluations.

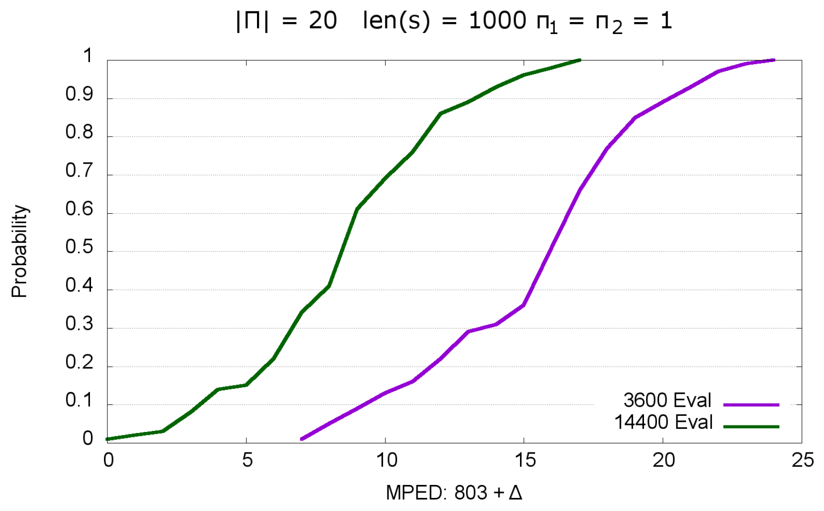


Figure 5-31: Probability distribution for a particular MPED value with $|\Pi| = 20, len(s) = 1000, \pi_1 = \pi_2 = 1$

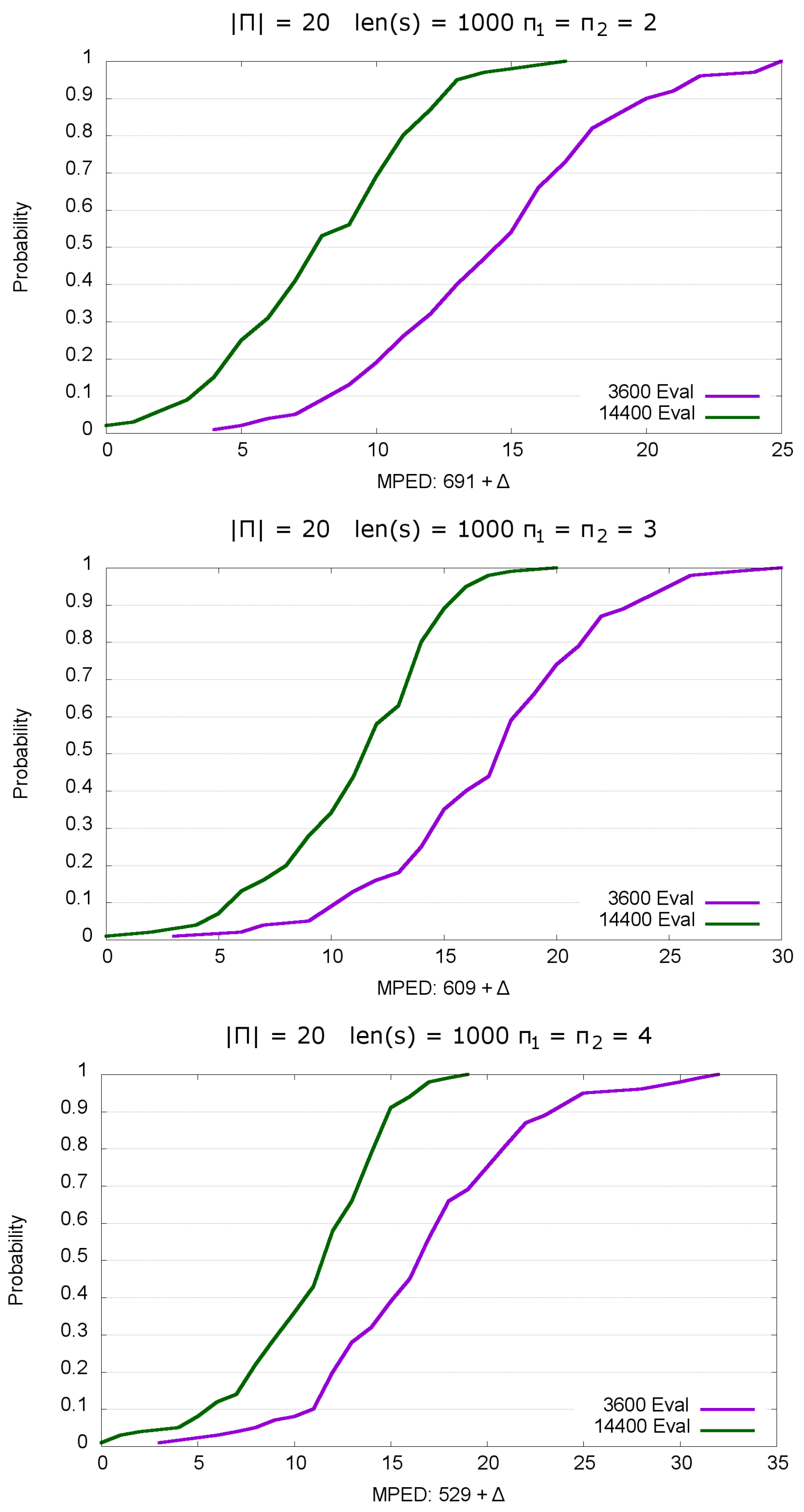


Figure 5-32: Probability distribution for a particular MPED value with $|\Pi| = 20, len(s) = 1000, \pi_1 = \pi_2 = \{2, 3, 4\}$

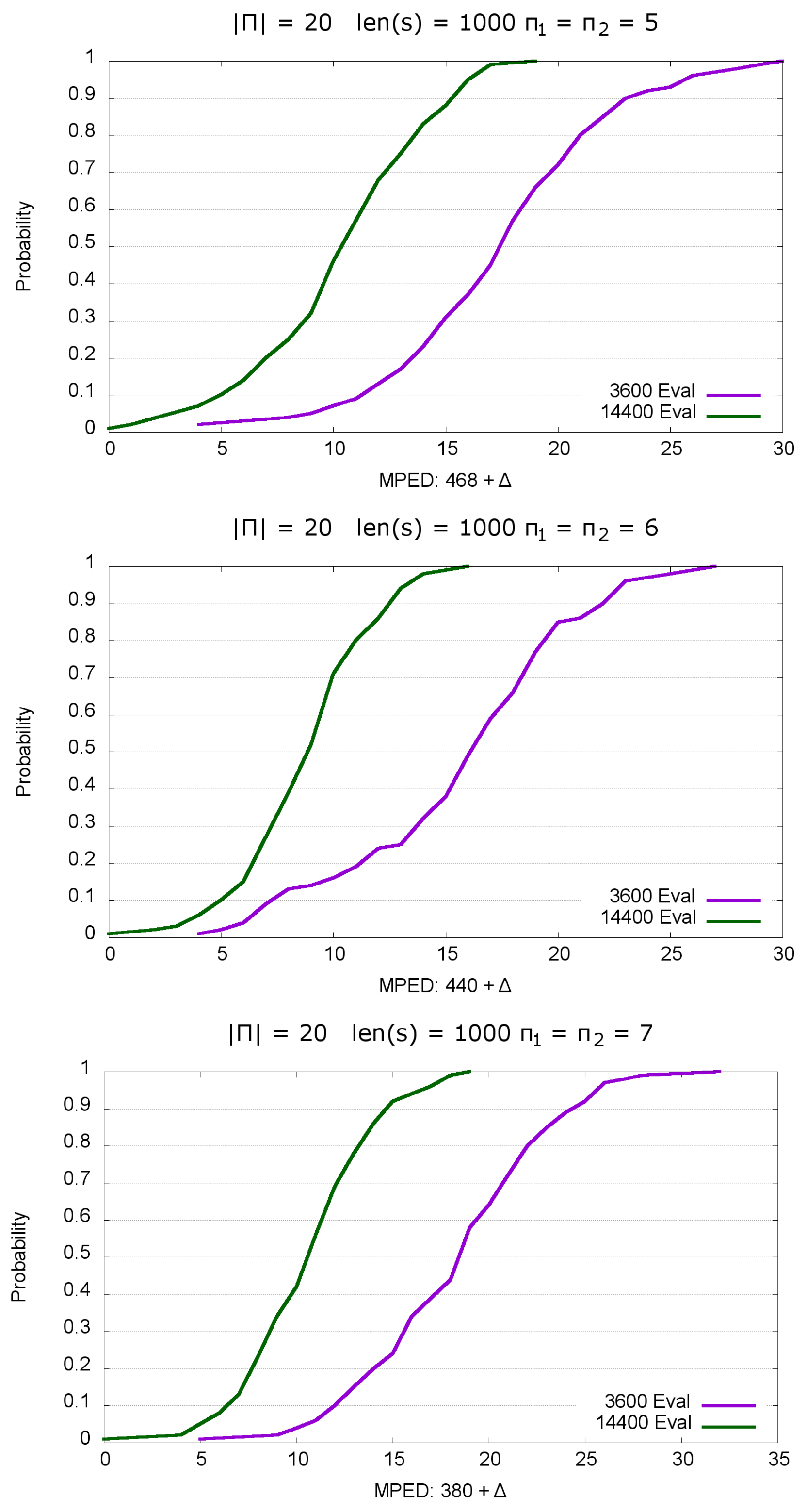
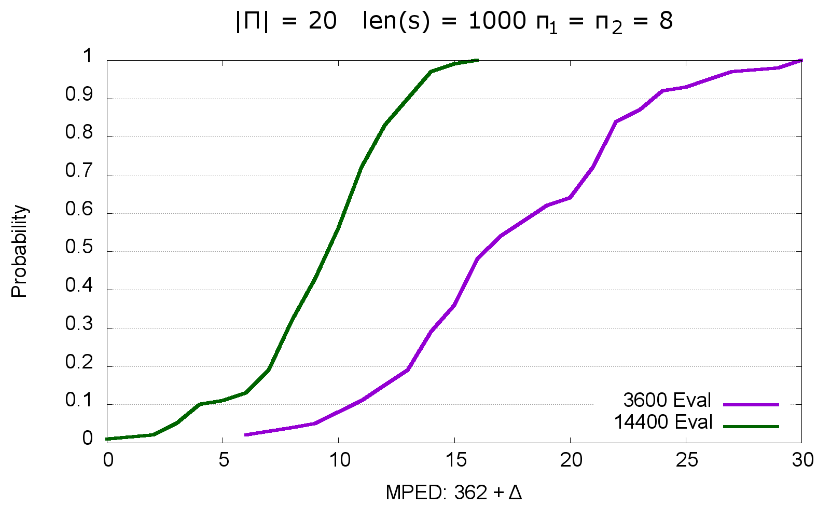
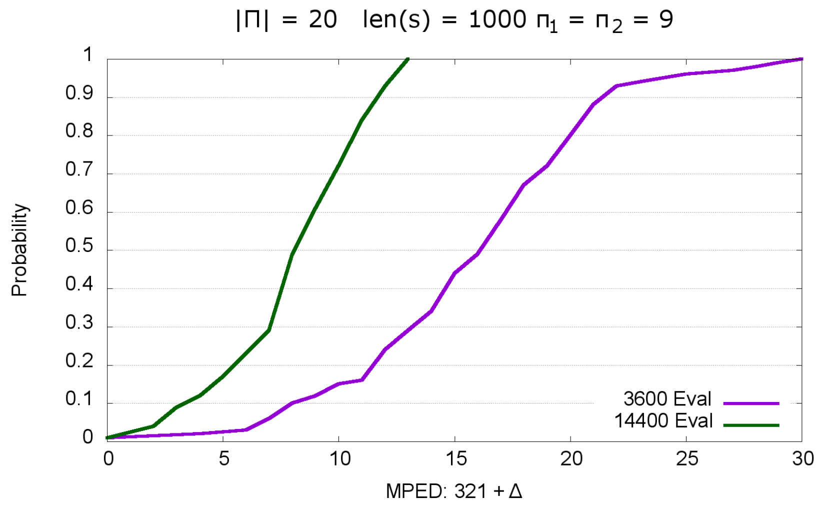


Figure 5-33: Probability distribution for a particular MPED value with $|\Pi| = 20, len(s) = 1000, \pi_1 = \pi_2 = \{5, 6, 7\}$





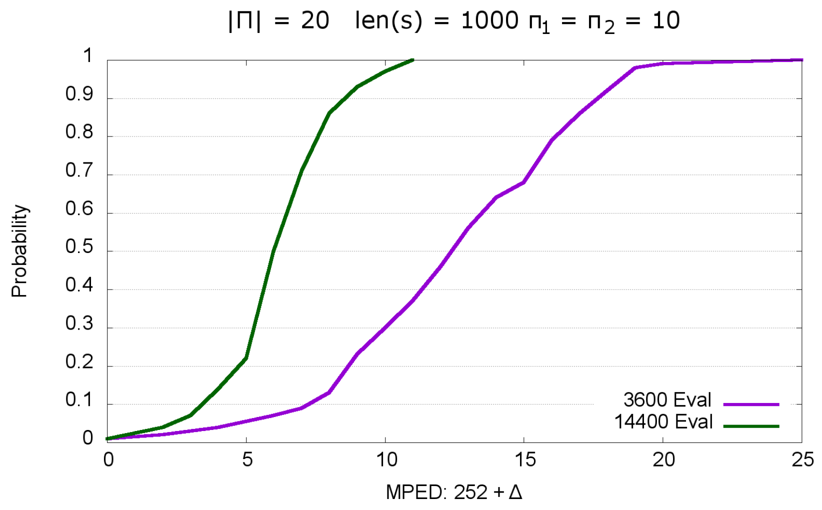


Figure 5-34: Probability distribution for a particular MPED value with $|\Pi| = 20, len(s) = 1000, \pi_1 = \pi_2 = \{8, 9, 10\}$

From the analysis of these figures we observe that both with 3600 and 14400 evaluations the probability of finding a value *near* the optimum is very high. Increasing the number of iterations obviously significantly increases such probability and, most importantly, reduces the range of values below the optimum returned by the evaluation. Finally, we can observe that 100% probability is always reached in less than 10% of the optimal value; this allows us to empirically conclude that the approach allows to guarantee high precision even with low numbers of iterations.

Chapter 6

Applications of the Multi-Parameterized Edit Distance

6.1 Introduction

This chapter is dedicated to show the application of the MPED in in different contexts, from strictly engineering ones, e.g., Wireless Sensor Area Networks, to medical based contexts such as White Matter Fiber-Bundles and Electroencephalography analysis. For each application context, we introduce the background and corresponding issues, and we provide a review of academic literature.

This thesis does not include all the considered application contexts (a complete overview can be found in the following papers [23–31, 35, 107]). Three main applications are described next, namely *(i) Wireless Sensor Area Networks*, *(ii) White Matter Fiber-Bundle* and *(iii) Electroencephalography* analysis.

In particular, Section 6.2 reports on Wireless Sensor Area Networks (WSANs) and related application. WSANs complexity is actually growing fast thanks to the Internet-of-Things world, thus various problems are arising. We concentrate on the problem of *anomaly detection*, i.e., the automatic detection of an anomaly in the network. MPED has been used in order to define a degree of similarity between time series of heterogeneous sensor data. Part of the work presented in Section 6.2 has been published in [23, 35].

Section 6.3 is dedicated to the context of White Matter Fiber-Bundles analysis. The objective here is reconstructing, visualizing and analyzing *in vivo* White Matter (WM) fibers.

Particular sets of fibers called *fiber-bundles*, representing different WM structures, connect different gray matter regions of the brain, allowing them to communicate. Thus, analyzing WM structures is a crucial step to understand and predict effects of some neurodegenerative pathologies. In this context, we defined a string-representation of the fibers and we used MPED in order to extract and characterize WM Fiber-Bundles. Part of the work presented in 6.3 has been published in [26–29, 107].

Finally, in Section 6.4, a new string-based and network-based approach is used to investigate neurological disorders. Electroencephalogram data are analyzed with a slightly different version of the MPED, called *consensus* MPED, in conjunction with network based analysis tools, to help experts in their work. In order to show the effectiveness of the approach, we show how it can be employed for Creutzfeldt-Jacob Disease, epilepsy and Alzheimer's Disease. Part of the work presented in Section 6.4 has been published in [25].

6.2 MPED for Wireless Sensor Area Networks

6.2.1 Background

Recent technological and software improvements allowed a widespread diffusion of Wireless Sensor Networks and related applications [3, 45]. This led to a significant increase in the amount of produced sensor data and in the complexity of sensor networks. Significant effort has been spent in the last few years on the definition of frameworks for a flexible and efficient management of Wireless Sensor and Actuator Networks (WSANs) [46, 108]; this includes intelligent sensing/actuation techniques, as well as data abstractions for improved data analysis.

The complexity of WSANs is constantly growing. In fact, a growing number of networks include heterogeneous sensors, i.e. devices producing different kinds of signals/measures/messages. As an example, sensors in the network may produce not only differently scaled real value data, but also text messages, discrete signals, symbolic alerts, etc.

While sensors network management and the development of robust data acquisition layers received much attention in the literature, one big open challenge in WSANs is anomaly detection [17, 18], i.e. the detection of unexpected behavior in incoming data. Anomalies can be generated either by malfunctioning in the sensors or by deviations in the environment. In most cases, it is a challenging task being able to distinguish between the two.

Most of the approaches for anomaly detection concentrate on the analysis of data produced by each *single* device [126]. This is mainly done by fairly complex mathematical analysis of data streams; however, these approaches can be usually applied on numerical data only. Other approaches compare incoming data packets to fixed patterns identifying known behavioral models [2]; typical applications of this kind of techniques are fraud detection for credit cards [94] and intrusion detection in security [47]. Neural networks are often applied also in this setting [18]. However, all these techniques are not well suited for heterogeneous sensor networks.

Due to the nature of the problem being intrinsically denoted by a heterogeneous kind of network, we proposed a novel approach specifically conceived for monitoring heterogeneous WSANs using MPED. In particular, our proposal is to identify (hidden) correlations between sensors and to exploit such knowledge to monitor the behaviour of sensors during their working life. As an example, assume we are able to identify that the behavior of two different and heterogeneous sensors is, for some not necessarily obvious reason, correlated. The observation of significant variations of this correlation during time may allow us to suspect that some anomaly is occurring. In fact, it may happen that for two sensors measuring light and temperature, values of temperature are actually influenced by (e.g., sun) light or vice versa (think for example to sensors near incandescent objects).

Obviously, this correlation measure should be robust enough to endorse reasonable variations like time shifts and value drifts, but to identify spikes and noises in the signals.

As previously pointed out, when dealing with heterogeneous data streams, generated sequences may come from very different contexts and may be represented with different symbols/metrics. Moreover, in order to accommodate possible drifts, it is necessary to provide the flexibility of mapping some symbol of the first sequence into more than one symbol of the second sequence (and vice versa) so that *many-to-many* mappings can be considered. Giving these considerations, MPED exactly fits these prerequisites.

We tested our approach in an experimental environment based on the Building Management Framework (BMF). Results show that the proposed approach is actually capable of identifying hidden correlations, is robust enough to acceptable environment variations and is capable to identify potential sensors faults. We note that to the best of our knowledge there is no approach in the literature facing all the problems outlined above.

This Section is organized in two parts: first we present the exploited case study and we give further information to understand it, then we describe our experiments and discuss corresponding results.

6.2.2 Case study

As a case study, ambient data have been sensed from a set of wireless sensor nodes deployed in an indoor building environment, specifically at DIMES, Department of Informatics, Modelling, Electronics and Systems, University of Calabria, as shown in Figure 6-1. In particular, the node tagged as *(a)* has been positioned in an air conditioned and artificially illuminated laboratory; node *(b)* has been located in a corridor without windows and with air conditioning system; node *(c)* has been placed in an office room far from the direct sunlight; nodes *(d)* and *(e)* have been placed both in the same room as node c, but with their sensors leant against a window.

Nodes organized in a multi-hop wireless sensor network have been effectively and efficiently managed through the Building Management Framework (BMF) [46]. The BMF is a domain-specific framework designed for the flexible and efficient management of WSANs deployed in buildings. It offers features such as fast prototyping of WSAN applications, intelligent sensing/actuation techniques, and abstractions for capturing the floor plan of a building.

BMF enables the use of heterogeneous WSANs managed by a basestation, which acts both as a network configurator and a data collector. Basestation and nodes communicate through the BMF Communication Protocol, an application level protocol built on top of multi-hop networks protocols (Dissemination and Collection Tree Protocols [51, 76]) An example of BMF network is shown in Figure 6-2, where the BMF high-level layered architecture for both the BMF basestation and node sides is shown. In particular, on the basestation-side, the BMF architecture is split in layers comprehending: (i) support for heterogeneous sensor platforms (e.g. TelosB, Tyndall, Shimmer, SunSPOT), (ii) a network management layer that allows to flexibly manage the BMF network through configuration packets sent over the air, (iii) a Basestation Core providing a set of functionalities to manage/configure the network (e.g. group nodes, create periodic sensing or actuation requests to the network), and (iv) a set of applications that can be run on top of the Basestation Core; on the node-side the BMF layers comprehend: (i) a set of platform-specific components to allow the use of different type of nodes in a BMF network, (ii) a network management

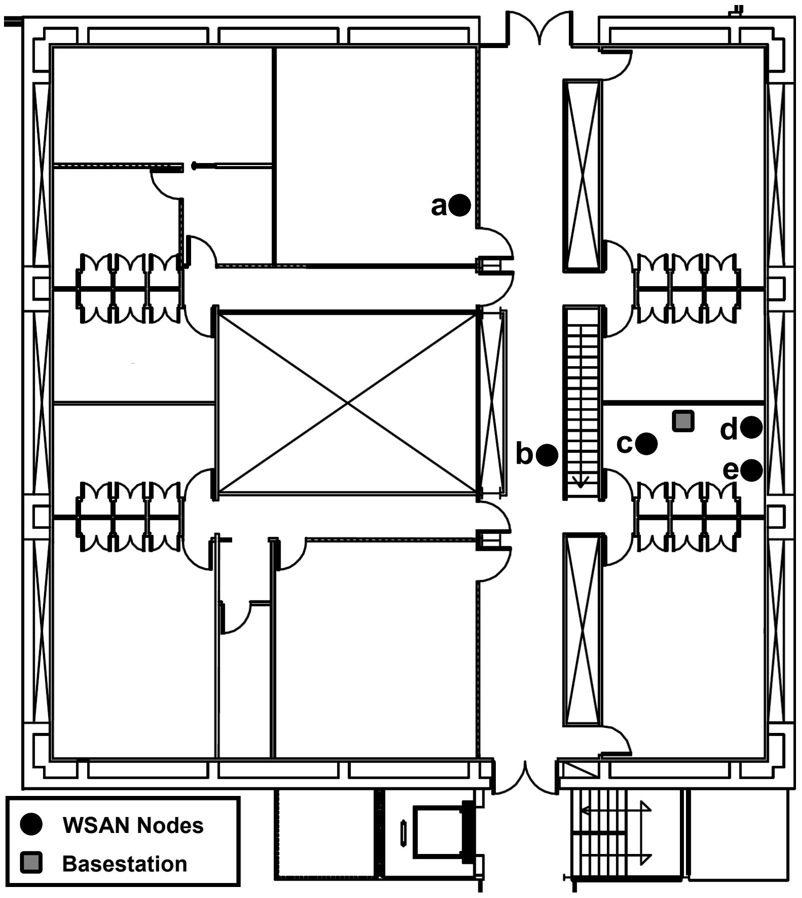


Figure 6-1: Wireless sensor nodes deployed at DIMES.

layer to allow communication among nodes and with the basestation, and (iii) a platform-independent core to implement the node specific functionalities, such as signal processing and multi-request scheduling on the nodes functionalities.

For the case study, BMF nodes have been configured to send to the basestation synthetic data every minute. In particular, every node in the deployment collects data from light and temperature sensors every second and every minute sends to the basestation the mean computed over the samples read. To provide the approach with a complete input, the BMF basestation has been improved with a filter that removes redundant packets received from the network and purposely masks data losses.

6.2.3 Experiments and discussion

In this section, we report results for a number of tests carried out to assess the effectiveness of our technique. Tests have been carried out collecting one whole day data from different

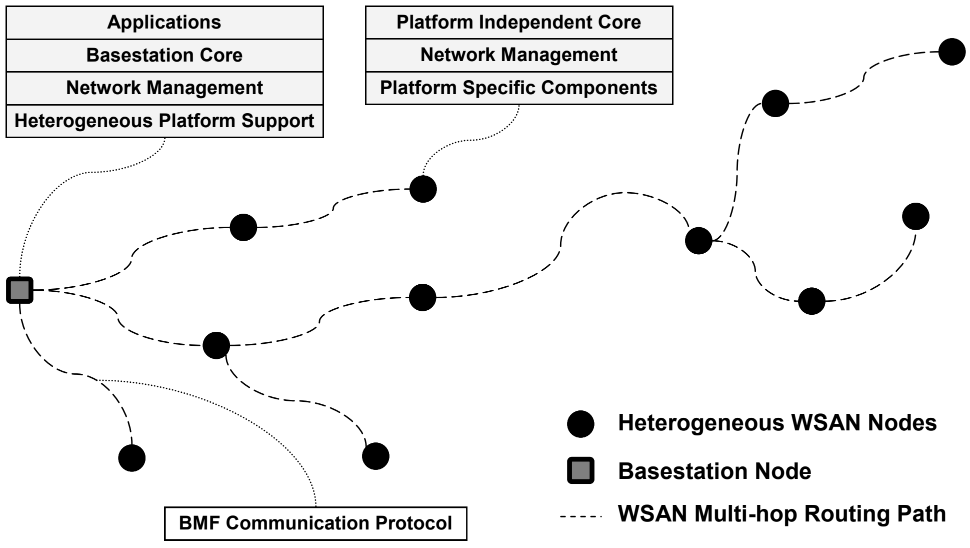

Applications
Basestation Core
Network Management
Heterogeneous Platform Support
Platform Independent Core
Network Management
Platform Specific Components
BMF Communication Protocol
Heterogeneous WSAN Nodes
Basestation Node
WSAN Multi-hop Routing Path

Figure 6-2: A BMF Network.

wireless sensor nodes (see Section 6.2.2).

Collected numerical data have been discretized in order to produce one string for each pair node-sensor; for each string s_i, $len(s_i) = 500$ and $\Pi_i = 20$. We then concentrated on comparing light and temperature data coming from each node. We carried out the experiments using MPED (with the ad-hoc hill climbing heuristic) and it has been executed on a server equipped with an Intel Xeon X3430 processor and 4 GB of RAM running the Ubuntu Linux kernel 2.6.26-2-686-bigmem SMP i686 GNU/Linux operating system.

We carried out three kinds of tests, which are detailed next.

6.2.4 Hidden correlation for different positioning of the sensors nodes

In this test, we considered only the nodes a-d. Figure 6-3 plots the raw data collected from considered nodes; observe that it is hard to state, from the figures only, some degree of correlation between measured temperature and light.

We first measured $I_{\langle \pi_1, \pi_2 \rangle}$ for the four sensors nodes and using different configurations, namely $I_{\langle 1,1 \rangle}$, $I_{\langle 2,2 \rangle}$, and $I_{\langle 3,3 \rangle}$; moreover, we computed the same measures on randomly generated string pairs (having the same lengths and alphabets as the test ones) and averaged obtained values. Results are shown in Table 6.1. From the analysis of this table, it is possible to observe that the most correlated measures are those obtained from nodes *(a)* and *(d)*. Obtained results confirmed our intuition for *(d)* since intuitively temperature

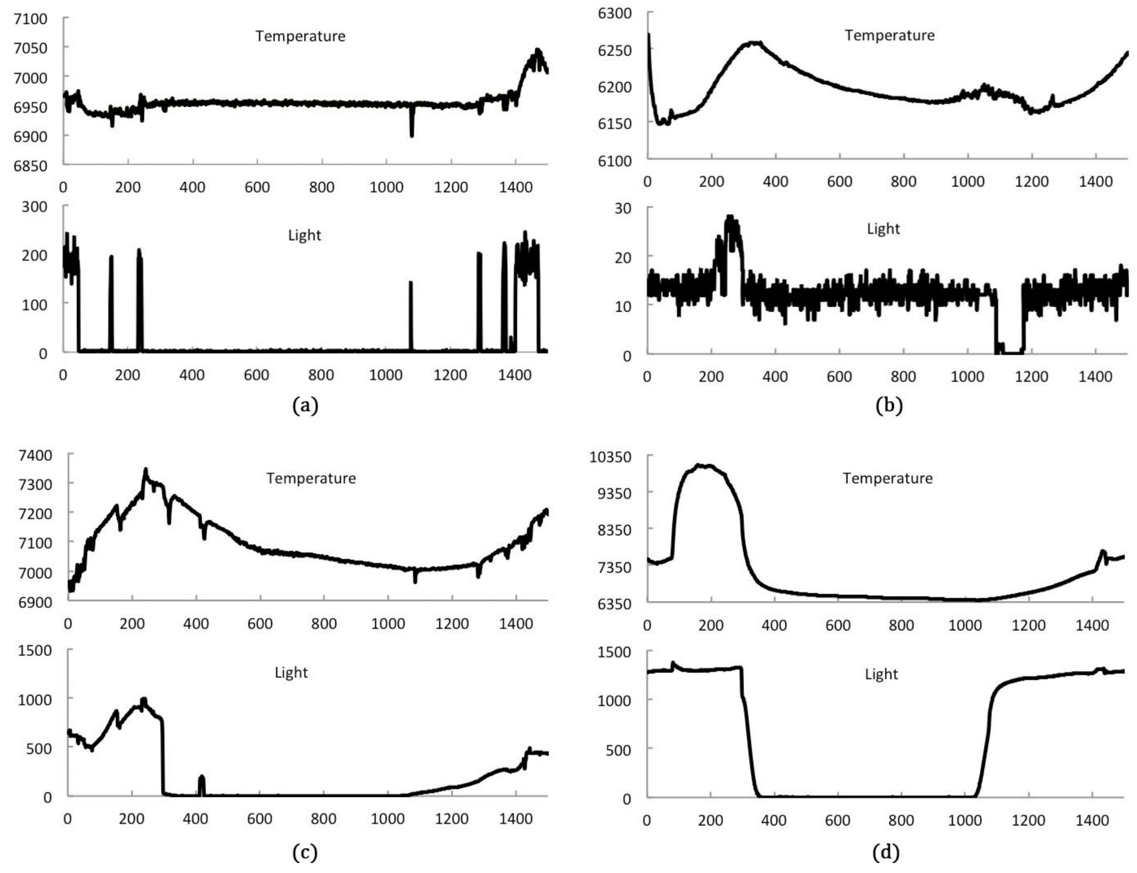


Figure 6-3: Plot of (T)emperature and (L)ight collected from nodes a,b,c,d

is significantly dependent on sunlight; however results for *(a)* where not so obvious but they can be motivated by the fact that temperature and light are kept almost constant by artificial illumination and conditioning (see Figure 6-3).

In order to have a comparison meter for these results, we also computed the standard mathematical correlation degree between numerical sequences (see Table 6.1), which basically confirmed the trends measured with MPED. Observe, however, that computing this measure is possible only between pairs of numerical data; as an example, we could have not computed it if one of the sensors produced labelled messages; on the contrary, $I_{\langle\pi_1,\pi_2\rangle}$ is a more general measure which may compare heterogeneous sequences, and can take into account both temporal and amplitude shifts.

Node	$I_{\langle 1,1\rangle}(T,L)$	$I_{\langle 2,2\rangle}(T,L)$	$I_{\langle 3,3\rangle}(T,L)$	Std Corr.
a	62.00%	75.00%	81.80%	0.49
b	30.40%	45.80%	54.40%	0.22
c	41.40%	52.60%	69.60%	0.54
d	55.50%	72.00%	80.80%	0.61
expected random	27,57%	41,80%	49,90%	0.016

Table 6.1: MPED and Std Correlation

Node	$I_{\langle 1,1\rangle}(T,L)$		$I_{\langle 2,2\rangle}(T,L)$		$I_{\langle 3,3\rangle}(T,L)$	
	Day 1	Day 2	Day 1	Day 2	Day 1	Day 2
d	51.72%	51.01%	64.94%	67.20%	72.77%	75.85%
e	49.04%	43.17%	63.54%	66.91%	71.67%	75.46%

Table 6.2: Day span

6.2.5 Robustness of the measure

In a second series of experiments, we verified the robustness of the approach to natural and artificial variations in the measurement context. Specifically, we considered nodes d-e from which we collected the stream of Temperature and Light for two consecutive days. Moreover, the second day of observation, one of the nodes has been covered with an opaque sheet in order to simulate a "cloudy" day. Obtained results for $I_{\langle \pi_1,\pi_2\rangle}$ are shown in Table 6.2. The analysis of this table shows that *(i)* correlation results remain stable throughout the days and that *(ii)* the proposed measure is robust to context variations. In fact, the correlation computed for node *(e)* does not significantly change over the two days even if this was the one covered by the opaque sheet.

6.2.6 Sensitivity to sensor faults

In the third series of experiments, we checked the sensitivity of the approach to possible faults of node sensors. In particular, we simulated faults in one of the sensors of a node introducing randomly generated out-of-scale noise in the stream. Then, we computed $I_{\langle \pi_1,\pi_2\rangle}$ for different percentage of noisy values. Obtained results are illustrated in Figure 6-4 where it is possible to observe that the correlation index correctly decreases for an increasing amount of noise. From the analysis of this graph it is possible to conclude that our approach could be possibly able to identify potential sensor faults when observing significant variations in the correlation index.

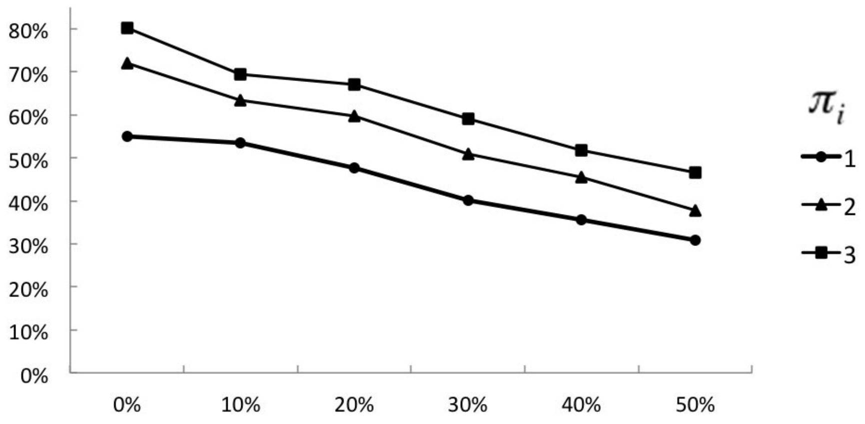


Figure 6-4: Sensitivity to sensor faults, Node *(d)*

6.3 MPED for White Matter Fiber-Bundles

6.3.1 Background

Reconstructing and visualizing *in vivo* White Matter (WM) fibers is a challenging issue in the investigation and the study of brain. For instance, the knowledge of these fibers is useful to understand and predict the effects of some neurodegenerative pathologies, like multiple sclerosis [91, 120]. Moreover, they can be used in neurosurgical planning to interactively guide the surgeon during an operation [52]. Right now, the most accurate method present in literature to perform this task is tractography [87], which is based on the analysis of the main diffusion directions of the water molecules when they move through WM tissues. This movement can be analyzed by a Magnetic Resonance Imaging (MRI) technique called Diffusion Tensor Imaging (DTI) [12], which has been used in our context. Also, other different methods for automated probabilistic reconstruction of sets of major white-matter pathways have been proposed [13, 14, 122]. From an anatomic point of view, particular sets of fibers called *fiber-bundles* represent different WM structures [22]. These connect different gray matter regions of the brain, allowing them to communicate. To analyze WM structures, it is crucial to isolate subsets of fibers belonging to the WM regions into consideration. This task is often performed manually by expert neuroanatomists that define inclusion and exclusion criteria in such a way to delineate regions of interests and isolate specific WM fiber-bundles [63, 82]. However, this way of proceeding is time consuming and it strongly depends from the operator. Moreover, another different issue represents an obstacle: the amount of data to analyze when facing this problem is enormous. As for an example, a simple whole brain tractography could generate up to 10^6 fibers. As a consequence, investigating WM structures of a cohort of people becomes a big data application and cannot be performed manually. In order to overcome this limitation, different automated approaches to extracting and, then, characterizing WM fiber-bundles have been proposed in the past literature (see, for instance, [49, 90, 121, 125]). WM fiber-bundles models, usually constructed by experts, provide a coincise representation of the bundles of interest. Then, given a set of models, the extraction and characterization of WM fiber-bundles from tractography data resorts to determining the real fiber-bundles corresponding to these models. A way to perform this last task consists of clustering WM fibers derived from tractography data and, then, applying a model-based characterization algorithm to identify fiber-bundles, starting from available models and obtained clusters. As will be clear below, this way of proceeding can guarantee the indispensable efficiency when the number of models to process is high. Nevertheless, in case where this approach is applied as it is, fibers should be clustered on the basis of their layout in the three-dimensional space. Therefore, we are in presence of a multi-dimensional clustering problem. This can be considered as a simplified version of the more complex multi-view clustering problem, which is well known for being a very difficult problem in the literature [19, 110].

6.3.2 Extract and characterize White Matter Fiber-Bundles

As a contribution in this setting we aim at proposing a new automated approach that, given as input a set of WM fibers, generated from streamlines produced by tractography, and a set of models, extracts fiber-bundles through clustering and a subsequent model-based characterization. The core "ingredients" of this proposal essentially are *(i)* a new string-based formalism allowing an alternative representation of WM fibers and *(ii)* a model-based characterization algorithm, fused together with the MPED.

Our approach can be summarized in four main steps, namely:

1. representation of WM fibers as strings;
2. computation of the dissimilarity degree, i.e., MPED value, for each pair of available WM fibers;
3. clustering of involved WM fibers;
4. exploitation of obtained clusters to identify the fiber-bundles corresponding to the models of interest.

Thanks to the string representation of the fibers, our WM fiber clustering task reduces to a string clustering one, which is simpler to face. Indeed, it is sufficient to find a metric to measure string dissimilarity, construct a dissimilarity matrix and apply a suitable clustering algorithm. In this case, the MPED optimally fits the problem and we will see how using the MPED leads to better results than a classical string dissimilarity metric, e.g., edit distance.

After the set of WM fiber clusters has been determined, our approach is able to exploit obtained clusters to extract the interesting fiber-bundles corresponding to the given models. Specifically, given a set of approximate models of fiber-bundles constructed by an expert, our approach matches each model m with a corresponding cluster and, then, exploits this matching to identify the WM fiber-bundles closest to m. Observe that, once clusters are available, the matching process can be carried out efficiently, because it is not necessary to compare each model with the whole set of available WM fibers. Indeed, it is sufficient to exploit medoids to characterize clusters and, consequently, to compare each model m with the medoids of extracted clusters to determine the one closest to m. Clearly, the number of medoids is generally orders of magnitude smaller than the number of WM fibers. This provides users with the possibility to dynamically change the models of interest and to immediately obtain the corresponding bundles.

String representation of WM fibers

The main purpose of this step is to represent a three-dimensional fiber in a different format, more compatible with clustering. In the past, several ways for representing a three dimensional line have been proposed. These different representations depend on both the context and the expected use.

In this context, we choose to represent a fiber as a sequence of voxels (volumetric picture elements), representing, in their turn, values on a grid in a three-dimensional space. By adopting this guideline, a generic fiber $f_i \in F$ can be represented as a sequence $f_i = (v_{i_1}, v_{i_2}, \ldots, v_{i_m})$ of voxels in the three-dimensional space. This sequence is aimed to approximate the corresponding three-dimensional line. As a consequence, the number of voxels representing a fiber is proportional to its length. Therefore, fibers having different lengths will be represented by a different number of voxels. The position of each voxel in the three-dimensional space is determined on the basis of the position of the neighboring voxels.

A color c_{i_j} is assigned to each voxel $v_{i_j} \in f_i$. This color is determined by the orientation of v_{i_j} in the three-dimensional space [38]. Specifically, let $p^x_{i_j}$ (resp., $p^y_{i_j}$, $p^z_{i_j}$) be the fraction of information of v_{i_j} parallel to the x (resp., y, z) axis. A basic color $\overline{c^x}$ (resp., $\overline{c^y}$, $\overline{c^z}$) is associated with the x (resp., y, z) axis. On the basis of the standard code of colors defined in [74], let $\overline{c^x}$ = red, $\overline{c^y}$ = green and $\overline{c^z}$ = blue. c_{i_j} is, then, obtained by the weighted combination of $\overline{c^x}$, $\overline{c^y}$ and $\overline{c^z}$. Clearly, adopted weights are strictly related to the orientation of v_{i_j} in the three-dimensional space. For instance, if v_{i_j} is parallel to the x axis, $p^x_{i_j} = 1$, $p^y_{i_j} = 0$, $p^z_{i_j} = 0$, and, consequently, $c_{i_j} = \overline{c^x} = red$.

Thanks to this notation, a fiber $f_i \in F$ can be expressed as a sequence $\phi_i = (c_{i_1}, c_{i_2}, \ldots, c_{i_m})$ of colors. Each element c_{i_j} corresponds to the color, expressed in the RGB color space, associated with v_{i_j}. Thanks to the fiber discretization tasks described above, a fiber $f_i \in F$, through its representation ϕ_i, can be easily translated into a string using a mapping function:

$$\Upsilon : RGB \rightarrow \Sigma$$

where $|\Sigma| = s$. In our approach, we decided to adopt the algorithm described in [7] for implementing Υ. Thanks to it, a fiber $f_i \in F$ can be expressed as a string in Σ^m.

The algorithm for implementing Υ belongs to the family of minimum variance quantization algorithms [59]. Quantization is a technique extensively used in image processing. It allows the reduction of the number of colors of an image. It clusters pixels into groups on the basis of the variance among the corresponding pixel values. In this way, it divides the RGB color cube into several smaller boxes and, then, maps all the colors falling within each box into the color of its center. In carrying out this task, it exploits the so-called RGB color cube, which is a three-dimensional array of all the colors that can be defined in the selected space. There are two main quantization methods proposed in the literature, namely uniform quantization and minimum variance quantization (which is the one adopted in our approach). They differ for the technique used to divide up the RGB color cube. The former cuts up the color cube into equally sized boxes; the latter divides the color cube into boxes of possibly different sizes, on the basis of the distribution of colors in the image. If the number s of boxes to obtain is an input parameter, then the algorithm automatically determines the position of boxes on the basis of the variance of the color data. Once the image is partitioned into s optimally located boxes, the pixels within each box are mapped to the pixel at the center of that box. Finally, a character is associated with each box center. In this way, an alphabet Σ, representing all the s boxes generated by the algorithm, is defined.

At the end of these activities, a set $F = \{f_1, f_2, \ldots, f_n\}$ of fibers can be represented as a set $T = \{t_1, t_2, \ldots, t_n\}$ of strings. Specifically, each element $t_i \in T$ is a string of Σ^m corresponding to f_i.

To formally express these transformations, we introduce a function $\tau(\cdot)$. First, it performs the transformation of f_i in a sequence of voxels; then, it associates a color with each voxel; finally, it transforms each voxel into a string. In the following, we use the notation $t_i = \tau(f_i)$ to represent the string t_i that the function $\tau(\cdot)$ returns when it takes f_i in input. The length of t_i is equal to the number of voxels exploited to represent f_i. As a consequence, if two fibers f_i and f_j have different lengths in the three-dimensional space, t_i and t_j have different lengths.

Algorithm Fibers-To-Strings (see Algorithm 7) describes the transformation of a set F of fibers into a set T of strings.

It receives a set $F = \{f_1, f_2, \ldots, f_n\}$ of WM fibers to transform, the cardinality s of the alphabet Σ which strings belong to, the size vxS of voxels in the three-dimensional space, and the size stS of the fiber step used by tractography algorithm. It returns a set $T = \{t_1, t_2, \ldots, t_n\}$ of strings representing the fibers in F.

The algorithm starts by computing the Step Rate stR, a parameter necessary for normalization in the next steps. For this purpose, it sets stR as the ratio of the step size stS of tractography to the norm of vxS. This computation of the norm of vxS is necessary because vxS is three-dimensional whereas stS is scalar. As a consequence, to perform the ratio, a scalar must be derived from vxS, representing it.

After the computation of stR, Fibers-To-Strings sets T to empty. Then, for each fiber $f_i \in F$, it performs some tasks devoted to obtain a string t_i over the alphabet Σ corresponding to f_i.

Specifically, t_i is initially empty. Then, on the basis of stS, a set $P_i = \{f_{i_0}, f_{i_1}, \ldots, f_{i_\nu}\}$ of three-dimensional points representing f_i are determined.

For each point f_{i_j}, $0 \leq j \leq \nu - 1$, of P_i, Fibers-To-Strings performs two tasks, namely:

- It determines the difference between f_{i_j} and $f_{i_{j+1}}$, which indicates the direction of the corresponding voxel in the three-dimensional space. Then, it normalizes this difference by dividing it by stR; let $p_i = (p_i^x, p_i^y, p_i^z)$ be the corresponding point in the three-dimensional space.

- It calls the function Υ, illustrated above, for transforming the point p_i in a character of the alphabet Σ. This character is then concatenated to t_i. The symbol $\oplus$ denotes string concatenation.

Once all the points f_{i_j}, $0 \leq j \leq \nu - 1$, of P_i have been processed, and t_i is complete, t_i

```
Input  : a set $F = \{f_1, f_2, \dots, f_n\}$ of fibers
         the cardinality $s$ of the alphabet $\Sigma$
         the size $vxS$ of voxels in the three-dimensional space
         the size $stS$ of the tractography step
Output: a set $T = \{t_1, t_2, \dots, t_n\}$ of strings
begin
    $stR = \frac{stS}{norm(vxS)}$;
    $T = \emptyset$;
    foreach $f_i \in F$ do
        $t_i = \emptyset$;
        $P_i = computeRepresentative(f_i)$;
        for $0 \leq j < |P_i|$ do
            $(p_i^x, p_i^y, p_i^z) = \left(\frac{f_{i_j}^x - f_{i_{j+1}}^x}{stR}, \frac{f_{i_j}^y - f_{i_{j+1}}^y}{stR}, \frac{f_{i_j}^z - f_{i_{j+1}}^z}{stR}\right)$;
            $t_i = t \oplus \Upsilon(p_i, s)$;
        end
        $T = T \cup t_i$;
    end
    return $T$;
end
```

Algorithm 7: ALGORITHM FIBERS-TO-STRINGS

is added to T.

FIBERS-TO-STRINGS terminates when all the fibers of F have been processed and, therefore, T is complete. It returns T as output.

Dissimilarity Matrix

The purpose of Step 2 is the construction of the Dissimilarity Matrix D associated with F. D is a $n \times n$ matrix; its generic element $d_{ij} = D[i, j]$ is a real number in the interval $[0, 1]$ and indicates the dissimilarity degree of the string representations t_i of f_i and t_j of f_j. To obtain the value of d_{ij}, it is necessary to apply a string-based dissimilarity metric on t_i and t_j. With regard to this issue, we point out that, in our application context, classical string-based dissimilarity metrics (like the Hamming or the Levenshtein distance), which measure the minimum number of edit operations necessary to transform the first string into the second one, would not work properly. As we pointed out in Chapter 1, they are based on the assumption that one-to-one correspondences between the symbols of the two strings are implicitly determined simply by identity.

Actually, in our scenario, the adoption of only one-to-one correspondences would be reductive and could lead to either imprecise or wrong results. In fact, in our application context, there could be different symbols expressing the same or similar concepts. Think, for instance, of two symbols, one representing a horizontal voxel and the other denoting a slightly oblique one, derived from an approximation during discretization: an error could arise if they are considered different. Analogously, it could be extremely important to be able to match a horizontal voxel with both another horizontal voxel and/or a slightly oblique one. Clearly, to avoid over-matchings, the number of these exceptions should be limited. Finally, there may exist pairs of symbols (e.g., a horizontal and a vertical voxel) that clearly should not match. In this case, it would be necessary to constrain invalid matches.

Thus, we use the MPED$_{SB}$ as a string dissimilarity metric due to the intrinsic consideration that, in this context, symbol identity should be considered together with the possibility of allowing many-to-many matchings and of specifying constraints.

The Dissimilarity Matrix D is thus computed as following: given a set $F = \{f_1, f_2, \ldots, f_n\}$ of WM fibers and the corresponding set $T = \{t_1, t_2, \ldots, t_n\}$ of strings, such that $t_i = \tau(f_i)$, given two integers π_1 and π_2 and a constraint χ, the generic element $D[i, j]$ of the Dissimilarity Matrix D associated with F is computed as:

$$D[i, j] = \mathcal{L}_{\langle \pi_1, \pi_2, \chi \rangle}(t_i, t_j) \tag{6.1}$$

Observations First, since the possible edit operations allowed during the computation of $\mathcal{L}_{\langle \pi_1, \pi_2, \chi \rangle}(s_1, s_2)$ are insertions, deletions and substitutions, the MPED$_{SB}$ allows the comparison of strings of different lengths. As for this specific case, not only our approach works properly even in presence of fibers of different lengths, but also it correctly returns a high dissimilarity value, in case of very different lengths, because a high number of insertions/deletions will be necessary. As a consequence of this fact, for instance, our approach is capable of recognizing as dissimilar two horizontal straight fibers having very different lengths, in spite of their identical direction in the three-dimensional space.

As a further consideration, we observe that spatial metrics require a fine-grained examination of the spatial coordinates of involved fibers. In fact, they either consider the Euclidean distance between fibers or require a fine-tuned registration of these last ones before starting distance computation. Euclidean-distance-based approaches may fail to identify similar fibers within thick bundles. In fact, if the proximity threshold is high, they may put together fibers with very different shapes; on the other side, if the proximity threshold is low, they may put in different bundles fibers approximatively having the same shape. Registration-based approaches are strongly dependent on the accuracy of the registration phase, which becomes a critical step. Our metric does not require any registration phase for comparing the fibers of a certain brain, and does not depend on spatial coordinates, since it exploits directionality information only. As a consequence, when applied to compute fiber dissimilarity, it is capable of overcoming the drawbacks of the two approach families mentioned above. As an example, fibers in a thick bundle will be all considered similar, since they approximatively share the same shape.

A further positive feature of MPED$_{SB}$ w.r.t. spatial metrics consists in its capability of avoiding the matching of two symbols representing totally different orientations (think, for instance, of a horizontal voxel and a strictly vertical one), which, again, is useful to discriminate similarities and dissimilarities of fibers despite their proximity in the three-dimensional space.

Last, but not the least, our MPED$_{SB}$-based approach is capable of supporting both expert and inexpert people, without the need of information regarding spatial locations.

Clustering of WM fibers

This step is aimed to pre-process F by applying a clustering algorithm on it in such a way as to group together anatomically homogeneous fibers. Thanks to the Dissimilarity Matrix D, computed during Step 2, the problem of clustering three-dimensional curves can be reduced to the one of clustering a set of strings, which can be faced by means of one of the many classical clustering algorithms already proposed in the literature. For instance, some clustering algorithms that can be easily incorporated in our approach are k-means [79], k-medoids [68], and Expectation Maximization - EM [37].

Algorithm WM-Fiber-Clusters (see Algorithm 8) describes the clustering task of our approach. It receives a set $F = \{f_1, f_2, \ldots, f_n\}$ of WM fibers and returns a set $Cl = \{cl_1, cl_2, \ldots, cl_k\}$ of clusters and the set $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_k\}$ of the corresponding medoids[1].

WM-Fiber-Clusters exploits the function $\tau(\cdot)$, described in Section 6.3.2, to implement *Step 1* of our approach, whereas it uses the first `foreach` cycle to implement *Step*

[1]Recall that the medoid γ of a cluster cl is the element of cl "least dissimilar" from all the other ones. Differently from other cluster representative elements (like mean), medoid is robust to noise and can be always determined, even when the context of interest does not support Euclidean distance.

```
Input  : a set $F = \{f_1, f_2, \ldots, f_n\}$ of fibers
Output: a set $Cl = \{cl_1, cl_2, \ldots, cl_k\}$ of clusters
         the set $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_k\}$ of the medoids of the clusters of $Cl$
begin
    foreach pair $(f_i,\ f_j)$ s.t. $f_i \in F,\ f_j \in F$ do
        $D[i, j] = \mathcal{L}_{\langle\pi_1,\pi_2,\chi\rangle}(\tau(f_i), \tau(f_j))$;
    end
    $Cl$ = cluster(F,D,$\kappa$);
    foreach $cl_j \in Cl$ do
        $Cl$=($Cl \setminus cl_j$) $\cup$ split($cl_j$);
    end
    $\Gamma$ = computeMedoids($Cl$);
    return $Cl$,$\Gamma$
end
```

Algorithm 8: Algorithm WM-Fiber-Clusters

2.

The clustering of the fibers of F is carried out by the function *cluster*, which implements the adopted clustering algorithm. The parameter κ is necessary if *cluster* implements a clustering algorithm requiring the number κ of desired clusters in input. The output of *cluster* is a set $Cl = \{cl_1, cl_2, \ldots, cl_\kappa\}$ of clusters; specifically, the cluster $cl_j \in Cl$ contains an anatomically homogeneous subset $cl_j = \{f_{j_1}, f_{j_2}, \ldots, f_{j_l}\}$ of the fibers of F.

The clusters of Cl obtained at the end of this step may still have a problem. Indeed, as pointed out by the observations gave at *Step 2*, MPED$_{SB}$ does not consider spatial information. As a consequence, it may happen that a cluster $cl_j \in Cl$ contains homogeneous fibers (i.e., all with similar shapes), but some of them actually distant in space (for instance, in different hemispheres). To face this issue, in our approach, each cluster cl_j undergoes a splitting phase aimed only to separate fibers very far in space. As far as this task is concerned, we point out that: *(i)* it does not require registration, because it compares fibers of the same brain; *(ii)* it is far less sensitive to space coordinates than purely spatial methods, since it is devoted to just identify very far fiber sets (for instance, fibers with homogeneous shape but belonging to different hemispheres). Function *split* carries out this task.

Finally, function *computeMedoids* returns the set $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_k\}$ of the medoids of the clusters of Cl. The medoid $\gamma_j \in \Gamma$ is used as the representative of the cluster $cl_j \in Cl$ and is exploited to speed up the model characterization phase in *Step 4*. In fact, as it will be clear in the next section, the computation of clusters can be considered as a pre-processing phase of our fiber-bundle extraction and characterization technique. This phase must be performed once and for all for each brain. As a consequence, once it has been carried

```
Input  : a set $Cl = \{cl_1, cl_2, \ldots, cl_k\}$ of clusters
         a set $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_k\}$ of the medoids of the clusters of $Cl$
         a set $\Psi = \{\psi_1, \psi_2, \ldots, \psi_q\}$ of models
Output: a set $\Theta = \{\theta_1, \theta_2, \ldots, \theta_q\}$ of bundles corresponding to $\Psi$
begin
    $\Theta = \emptyset$;
    foreach $\psi_l \in \Psi$ do
        $\Delta_l = \emptyset$;
        foreach $\gamma_j \in \Gamma$ do
            $\Delta_l = \Delta_l \cup \mathcal{L}_{\langle \pi_1, \pi_2, \chi \rangle}(\tau(\psi_l), \tau(\gamma_j))$;
        end
        $\mu = argmin(\Delta_l)$;
        $\Theta = \Theta \cup cl_\mu$;
    end
    return $\Theta$
end
```

Algorithm 9: Algorithm WM-Fiber-Bundles

out, the derivation of the fiber-bundles associated with each model $\psi_l \in \Psi$ requires the examination of only k medoids, instead of the n available WM fibers. Interestingly, k is, generally, orders of magnitude smaller than n.

Model-based WM fiber-bundles extraction and characterization

Once clusters and their medoids are available, our approach can perform the fiber-bundle extraction and characterization activities. For this purpose, it carries out *Step 4*, which, essentially, consists of the Algorithm WM-Fiber-Bundles (see Algorithm 9).

This algorithm receives a set $Cl = \{cl_1, cl_2, \ldots, cl_k\}$ of clusters, the set $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_k\}$ of the medoids of the clusters of Cl, and a set $\Psi = \{\psi_1, \psi_2, \ldots, \psi_q\}$ of models. Each model $\psi_l \in \Psi$ represents the approximate shape of a fiber-bundle of interest. It could be obtained in two different ways, namely: *(i)* by exploiting a spline curve to draw the profile of the fiber-bundle of interest; *(ii)* by importing the mean-line profile of the fiber-bundle of interest from an atlas of pre-labeled fiber-bundles. Interestingly, the approximate model adopted to extract a specific fiber-bundle may be adopted also to extract the same fiber-bundle from other images characterized by different resolutions, and possibly acquired from other subjects. Indeed, the representation of our model is based on its shape and is independent of its spatial location.

As previously pointed out, our approach does not require a complex registration phase. In fact, it only needs a smooth alignment of the reference axes of both the models and the brain. This because MPED$_{SB}$ disallows the matching of lines having completely different directions (such as a straight horizontal line and a straight vertical one), but allows the matching of lines with slightly different directions (such as a horizontal line and a slightly oblique one).

The output of WM-Fiber-Bundles is a set $\Theta = \{\theta_1, \theta_2, \ldots, \theta_q\}$ of derived fiber-bundles such that each $\theta_l \in \Theta$ contains a subset of the fibers of F.

In more detail, for each model $\psi_l \in \Psi$, WM-Fiber-Bundles generates the corresponding fiber-bundle by associating the most appropriate cluster of Cl with it. For this purpose, it constructs the set Δ_l of the distances between each medoid of Γ and ψ_l. In order to compute the distance between ψ_l and a medoid $\gamma_j \in \Gamma$, WM-Fiber-Bundles transforms both of them into two strings $s_l = \tau(\psi_l)$ and $s_j = \tau(\gamma_j)$ by applying the function $\tau(\cdot)$, described in Section 6.3.2. Then, it computes the MPED$_{SB}$ distance between s_l and s_j. After this, it determines the minimum value δ_μ of Δ_l. Clearly, δ_μ corresponds to a medoid $\gamma_\mu \in \Gamma$ and to a cluster $cl_\mu \in Cl$. Finally, it identifies cl_μ as the cluster of WM fibers having the closest structure and features to ψ_l and returns cl_μ as the fiber-bundle corresponding to ψ_l.

We point out, again, that, thanks to the pre-processing clustering phase, this step requires the examination of only k medoids, instead of the n available WM fibers. This feature, and the consequent efficiency of our approach, make it possible to easily apply the same models to different subjects (and, consequently, to the results of different tractographies) in a reasonable time. This way, it is possible, for instance, to compare the fiber-bundles of different people, such as healthy and ill patients.

Experiments

In this section we present the experimental campaign we carried out on the problems previously pointed out. First, we present tests of the pre-processing phase and the results of the application of MPED$_{SB}$. Then, we present some tests analyzing the computational efficiency. Finally, we present a real case study and corresponding tests.

Pre-processing phase As previously pointed out, the pre-processing phase of our approach includes the string-based representation and the clustering of available fibers. To validate this phase (measuring its effectiveness and efficiency), we performed several tests on a virtual phantom created by the phantom generator described in [21]. Figure 6-5 graphically displays this phantom. We asked an expert to manually annotate the corresponding

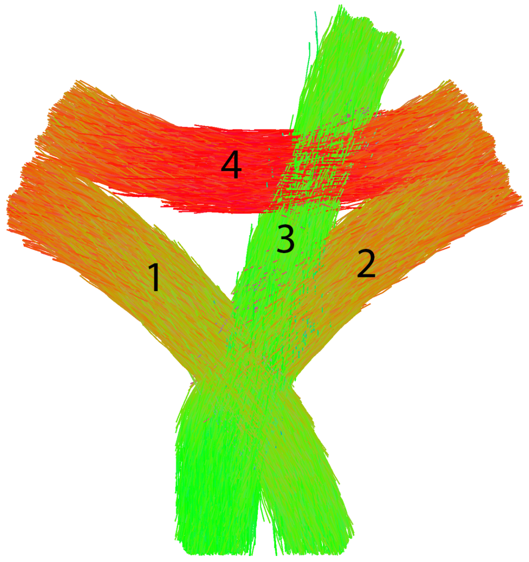


Figure 6-5: The virtual phantom used for our experimental campaign

data. Annotations performed by him are represented in the same figure; in particular, a number is associated with each bundle. As shown in Figure 6-5, the expert defined four fiber-bundles, namely: *(1-2)* diagonal, *(3)* vertical, and *(4)* horizontal bundles, which were used as a gold standard[2]. Indeed, our validation task required a ground truth and, in our opinion, the best possible ground truth was represented by a set of fiber-bundles manually annotated by an expert.

To perform our test activity, we compared the results of four approaches, namely: *(i)* our approach, with the usage of k-means as clustering algorithm in Step 3; *(ii)* our approach, with the exploitation of EM as clustering algorithm; *(iii) QuickBundles*, applied directly on the phantom; *(iv)* a baseline feature-based k-means, applied directly on the string-based representation of the phantom.

Whenever necessary (in particular, for cases *(i)*, *(ii)*, and *(iv)*), fibers were transformed into strings by applying the technique presented in Section 6.3.2; each fiber turned out to be characterized by 30 voxels. We set the cardinality of the string alphabet to 6. As for cases *(i)* and *(iv)*, we set the number of required clusters (i.e., k) to 4. As for cases *(i)* and *(ii)*, we set $\pi_1 = 1$ and $\pi_2 = 1$ for MPED$_{SB}$ and specified a constraint set χ aimed to avoid, for instance, matches between horizontal and vertical voxels. As for case *(iii)*, we run QuickBundles with a threshold value equal to $6mm$, which empirically was proved to

[2]Recall that, in statistics, the gold standard test refers to the most accurate test that can be carried out without restrictions.

	Precision	**Recall**	**F-measure**	**Overall**
Our approach - k-means	0.72	0.98	0.83	0.60
Our approach - EM	0.77	0.89	0.83	0.63
QuickBundles	0.91	0.55	0.69	0.50
K-means Baseline	0.44	0.44	0.44	-0.12

Table 6.3: Results for Cluster 1

	Precision	**Recall**	**F-measure**	**Overall**
Our approach - k-means	0.75	0.97	0.85	0.65
Our approach - EM	0.76	0.92	0.83	0.63
QuickBundles	0.94	0.51	0.66	0.48
K-means Baseline	0.47	0.41	0.44	-0.05

Table 6.4: Results for Cluster 2

	Precision	**Recall**	**F-measure**	**Overall**
Our approach - k-means	0.92	0.32	0.48	0.29
Our approach - EM	0.70	0.51	0.59	0.30
QuickBundles	0.51	0.78	0.62	0.04
K-means Baseline	0.48	0.35	0.40	-0.03

Table 6.5: Results for Cluster 3

	Precision	**Recall**	**F-measure**	**Overall**
Our approach - k-means	0.93	0.99	0.96	0.92
Our approach - EM	0.98	0.94	0.96	0.92
QuickBundles	0.82	0.99	0.90	0.78
K-means Baseline	0.33	0.64	0.44	-0.63

Table 6.6: Results for Cluster 4

produce the best results. Finally, as for *(iv)*, we considered each of the 30 voxels of a fiber as a feature of the fiber itself. Therefore, we defined the input of the feature-based clustering algorithm as a set of feature vectors $w_i = (x_{i_1}, \ldots, x_{i_m})$, where each w_i corresponded to a fiber f_i and each x_{i_j} corresponded to a voxel v_{i_j} of f_i. The approach delineated by *(iv)* can be used as a baseline of comparison because the only pre-processing task it requires consists of the transformation of fibers into strings. Therefore, it is lightweight and fast, and, if it provided satisfying results, it should be preferred to the other ones.

To evaluate the accuracy of all the four approaches, we compared the results obtained by them with the expert's annotation in such a way as to compute Precision, Recall, F-measure and Overall [96]. Obtained results are reported in Tables 6.3 – 6.6. Furthermore, qualitative and graphical representations of the clusters computed by cases *(i)* and *(iii)* are shown in Figures 6-6 and 6-7, respectively.

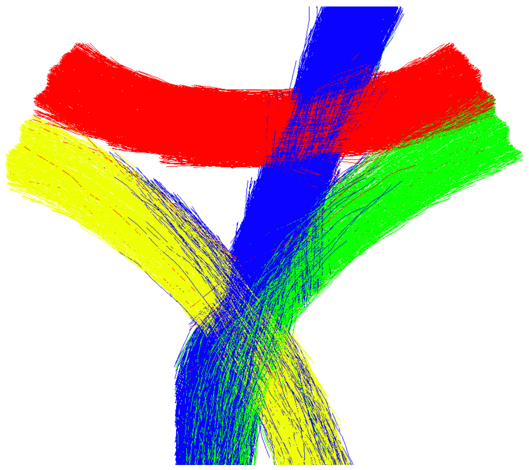


Figure 6-6: The clusters generated by our approach with the adoption of k-means as clustering algorithm

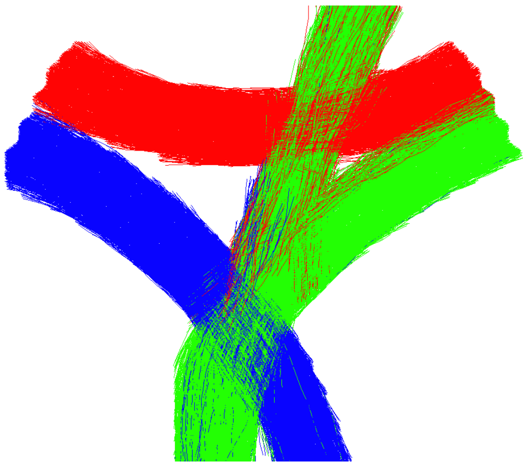


Figure 6-7: The clusters generated by QuickBundles

From the analysis of Tables 6.3 – 6.6, we can draw the following conclusions:

- The baseline approach is light and quick; however, its accuracy is so low to make it inapplicable in our context. In more detail, the values of Precision, Recall, F-measure and Overall returned by it are worse than the ones of all the other approaches and, in any case, are unsatisfying for all clusters.

- QuickBundles guarantees a very high Precision only for Clusters 1 and 2. As for these clusters, it obtains the best Precision among the four approaches into consideration. However, this result is obtained at the price of having a very low Recall. This behavior is caused by the fact that QuickBundles ignores information about voxel directionality and considers only voxel proximity. The same reasons represent the cause of the low value of Precision obtained by QuickBundles for Cluster 3. By contrast, QuickBundles shows a high value of Recall for this last cluster, which, if related with the low value of Recall for Clusters 1 and 2, testifies the difficulty of this approach to distinguish among bundles near in space.

- If we focus on cases *(i)* and *(ii)*, which correspond to two versions of our approach, we cannot observe a substantial difference between the results obtained by applying k-means and those returned by adopting EM. This fact represents a proof of the good robustness degree characterizing MPED$_{SB}$.

- Results obtained for all the clusters testify that the Precision of our approach is generally higher than the one of the other approaches. At the same time, our approach can guarantee a satisfying level of completeness testified by a satisfying value of Recall. The satisfying results of our approach are further confirmed by the analysis of F-measure and Overall. In fact, QuickBundles returns a higher F-measure than the one obtained by our approach only for Cluster 3. However, for this cluster, as shown above, the Precision of QuickBundles is unacceptable, and this fact produces a very low value of Overall.

Summarizing, the previous test shows that our approach can guarantee a good balance between Precision and Recall in the extraction and characterization of fiber bundles when directionality information plays a key role.

MPED$_{SB}$ metric In this test, we validated our approach's capability of characterizing WM fiber-bundles. The input dataset consisted of a virtual diffusion MR phantom, generated by Phantomas [21]. This phantom accurately simulated the brain complexity with the fiber geometry used in the 2nd HARDI Reconstruction Challenge (ISBI 2013). The ground truth was obtained by requiring our expert to manually segment all the fiber-bundles in the phantom. In this way, the approximate shapes of these fiber-bundles were defined and 17 models were identified. These are illustrated in Figure 6-8. In this figure, each bundle is colored on the basis of the standard code of colors for tractography defined in [74] and described in Section 6.3.2.

To perform our validation, we measured the distance between each of the 17 models and the fibers in the phantom. To carry out this task, we applied both MPED$_{SB}$ and the classic

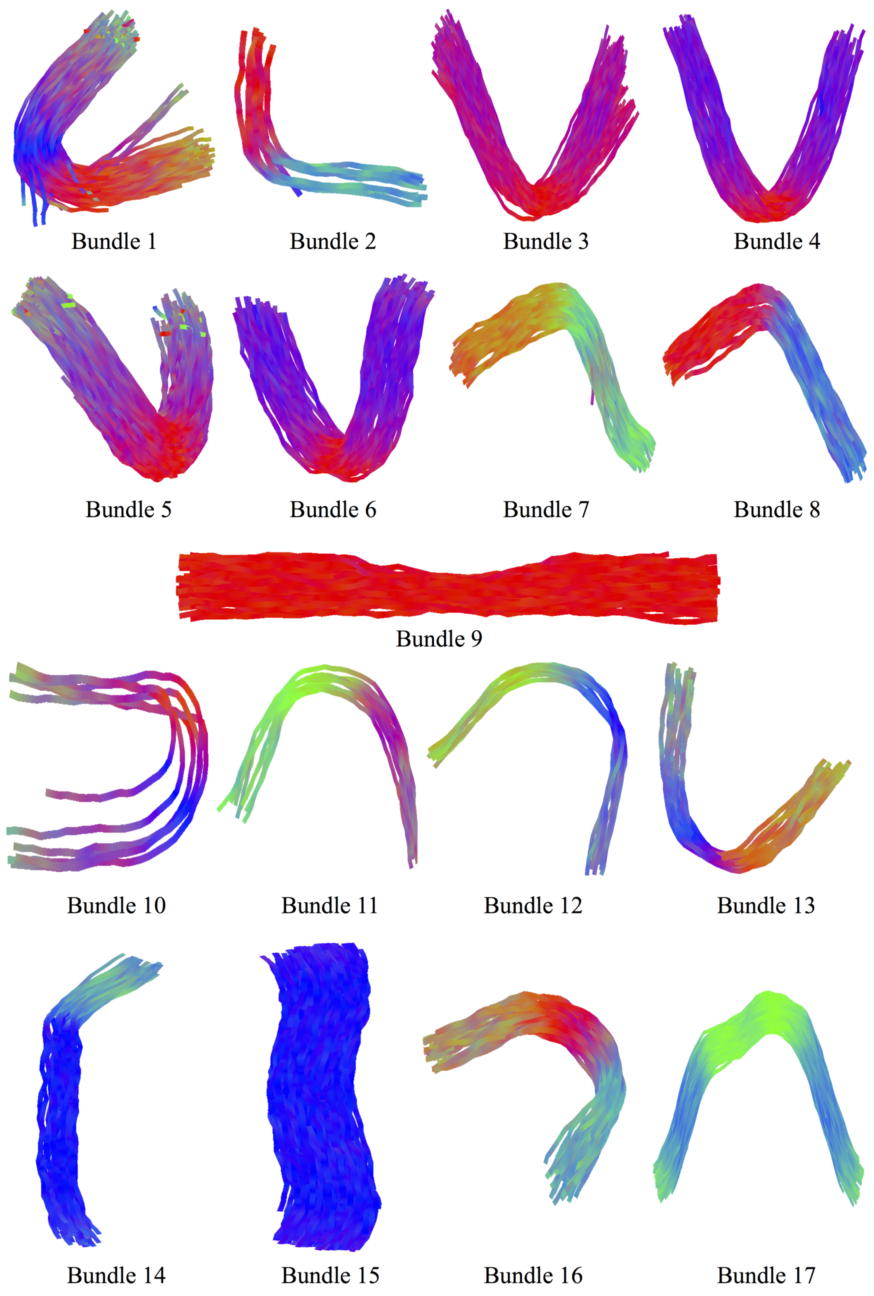


Figure 6-8: The 17 bundles identified in the diffusion MR phantom adopted in our test

edit distance. We compared each obtained result with the ground truth and computed Precision, Recall, F-measure and Overall [96] for both SBED and the edit distance. In Table 6.7, we illustrate the results obtained for each model, whereas, in Table 6.8, we present the average values of Precision, Recall, F-measure and Overall for the two distances. Before illustrating obtained results, we must preliminarily observe that, in our reference context, Precision is more important than Recall because the number of fibers generated

		Model							
		1	2	3	4	5	6	7	8
MPED$_{SB}$	Precision	1	1	0.2081	0.4069	1	0.5194	0.9412	0.9474
	Recall	0.0220	0.1111	0.7701	0.6344	0.4886	0.6505	0.3556	0.3462
	F-measure	0.0430	0.2000	0.3277	0.4958	0.6565	0.5776	0.5162	0.5071
	Overall	0.0220	0.1111	−2.1604	−0.2903	0.4886	0.0486	0.3334	0.3270
Edit	Precision	0	0	0.3602	0.4143	0	0.5276	1	1
	Recall	0	0	0.7701	0.6236	0	0.6505	0.3556	0.3270
	F-measure	NULL	NULL	0.4908	0.4978	NULL	0.5826	0.5246	0.4928
	Overall	NULL	NULL	−0.5978	−0.2580	NULL	0.0681	0.3556	0.3270

		9	10	11	12	13	14	15	16	17
MPED$_{SB}$	Precision	0.7333	0.6689	1	1	1	1	0.9797	1	0
	Recall	0.9245	0.6689	0.0577	0.0198	0.1154	0.0155	0.7472	0.2553	0
	F-measure	0.8179	0.6689	0.1091	0.0388	0.2069	0.0305	0.8478	0.4068	NULL
	Overall	0.5883	0.3378	0.0577	0.0198	0, 1154	0.0155	0.7317	0.2553	NULL
Edit	Precision	0.7333	0.6972	1	0	0	1	0.9797	1	0
	Recall	0.9245	0.6689	0.0192	0	0	0.0199	0.7472	0.1064	0.0007
	F-measure	0.8179	0.6828	0.0377	NULL	NULL	0.0390	0.8478	0.1923	NULL
	Overall	0.5883	0.3784	0.0192	NULL	NULL	0.0199	0.7317	0.1064	NULL

Table 6.7: Results obtained by applying MPED$_{SB}$ and the edit distance on the 17 models into consideration

	Avg. Precision	**Avg. Recall**	**Avg. F-measure**	**Avg. Overall**
MPED$_{\text{SB}}$	0.7885	0.3637	0.4032	0.3346
Edit Distance	0.5124	0.3067	0.3063	0.1581

Table 6.8: Comparison of the average Precision, Recall, F-measure and Overall obtained by applying MPED$_{SB}$ and the edit distance

by tractography algorithms usually does not reflect the number of real fibers of a human brain.

From the analysis of obtained results, it is possible to draw the following conclusions:

- MPED$_{SB}$ reaches a very high average Precision (i.e., 78.85%).
- It also reaches a satisfying average Recall (i.e., 36.37%) with peaks of 92%.
- The average Precision, Recall, F-measure and Overall of MPED$_{SB}$ are higher than the ones of the edit distance (i.e., +27.60%, +5.70%, +9.69%, +17.66%, respectively).
- There are few models (i.e., 3 and 17), where both MPED$_{SB}$ and the edit distance do not work properly. These cases need deep analyses in the future, which, probably, will lead to perform some corrections on both approaches.

All the previous results allow us to conclude that the adoption of the new metric MPED_{SB} (which, as specified above, represents one of the main contributions of this paper) represents a step forward in the computation of string similarity and dissimilarity.

Speed test In carrying out this test, we started from the following considerations: *(i)* the pre-processing phase of our approach must be carried out once and for all for each brain; as a consequence, it is possible to exclude it from our test; *(ii)* one of the quickest approaches for the extraction of WM fiber-bundles is QuickBundles; as a consequence, it appears reasonable to compare our approach with it.

We applied both our approach (after the pre-processing phase) and QuickBundles to the virtual diffusion MR phantoms shown in Figures 6-5 and 6-8.

At the end of these tests, we obtained that, on average, our approach showed to be 2.54 times slower than QuickBundles. For instance, to process the phantom of Figure 6-8, QuickBundles took 0.87 seconds, whereas our approach needed 2.21 seconds.

However, in our opinion, with regard to these results, two considerations are in order. In fact: *(i)* even if slower than QuickBundles, our approach show a quickness acceptable for real cases; *(ii)* as previously pointed out, QuickBundles has an important limitation in that it is incapable of distinguishing among bundles near in space; our approach overcomes this limitation, as testified by the higher values of accuracy measures obtained in the previous tests.

As a consequence, it appears reasonable to partially sacrifice quickness in favor of accuracy (and, therefore, to choose our approach instead of QuickBundles) in all those cases in which the accuracy of results is the most important feature.

Application on a control subject Till now, we carried out our tests on virtual phantoms. Now, it is time to test our approach as a whole and on a real case. For carrying out this task, we required the collaboration of a healthy volunteer, on whom our approach could have been applied. This volunteer underwent a MR examination on a 3 Teslas Siemens Prisma MR System (64 channels head-coil). Diffusion protocol consisted on the acquisition of 100 slices in the AC-PC plane, TR/TE = 13700/69 ms, FOV = 160×160, with a spatial resolution of 1.5 mm^3 along 45 gradient directions ($b = 3000\ s.mm^{-2}$). The Orientation Distribution Function (ODF) and the probabilistic tractography were computed using the algorithms of MRtrix [111].

We required our expert to draw approximate shapes in such a way as to extract two fiber bundles. These were Corpus Callosum (CC) forceps minor - Figure 6-9(a) - and right Cortico-Spinal Tract (CST) - Figure 6-10(a).

To carry out our validation task, first we transformed the available tractography fibers into strings. Then, we computed the Dissimilarity Matrix and performed the clustering activity (in particular, we chose EM as clustering algorithm). Finally, we carried out the extraction of fiber-bundles and, next, their characterization.

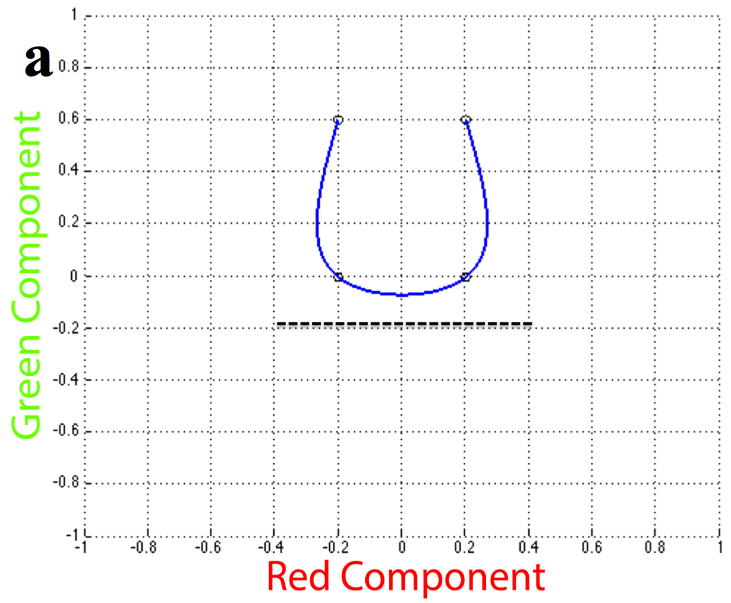
The extracted fibers are shown in Figure 6-9(b) and 6-10(b). An anatomical analysis of these figures is already sufficient to verify that our approach was capable of well extracting both forceps minor of CC (Figure 6-9(b)) and right CST (Figure 6-10(b)).

Discussion

After having described our approach, we want to point out that it can be easily extended from fiber-bundle extraction and characterization to several other contexts in which it is necessary to perform multi-dimensional (and, more in general, multi-view) clustering and characterization activities and/or in tasks requiring the integration of data belonging to different domains. In fact, it is sufficient to associate a color with each axis of the corresponding multi-dimensional domain to suitably color the corresponding voxel, and, after all voxels have been colored, to suitably discretize the corresponding color representation.

For instance, our approach can be adopted in all those biomedical contexts in which it is necessary to perform multi-dimensional (and, more in general, multi-view) clustering and characterization activities. Moreover, it can be adopted in those contexts someway requiring a multi-view clustering and a possible model-guided characterization of obtained clusters. As an example, in the clinical observation of the vital parameters of a patient [88], it is possible to simultaneously consider several measures, such as electrocardiogram, temperature, respiratory rate, etc. Thanks to our approach, all these measures can be analyzed simultaneously and compared in such a way as to evidence possible correlations.

Furthermore, our approach could be adopted also in contexts very different from the biomedical ones, whenever multi-view clustering and characterization tasks must be carried out. Think, for instance, of the analysis of air flows or of weather perturbation in meteorology or to the discovery of hidden correlations in multi-sensor (and possibly heterogeneous) a



b

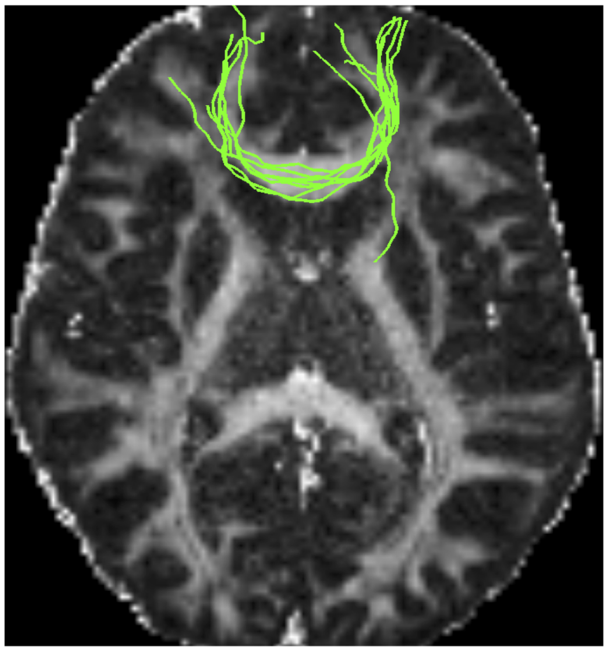


Figure 6-9: **a.** Approximate shape of Corpus Callosum (CC) and its axis of symmetry (black dotted line) drew by the operator; **b.** Extracted forcep minor of CC fibers (green)

data streams related to a unique phenomenon.

Finally, the general philosophy underlying our approach can be extended to other multi-view data applications. As an example, the search of frequent structures in a multi-dimensional space can be reduced to the search of frequent patterns in a set of strings. An analogous consideration holds for the search of specific structures in a multi-dimensional space.

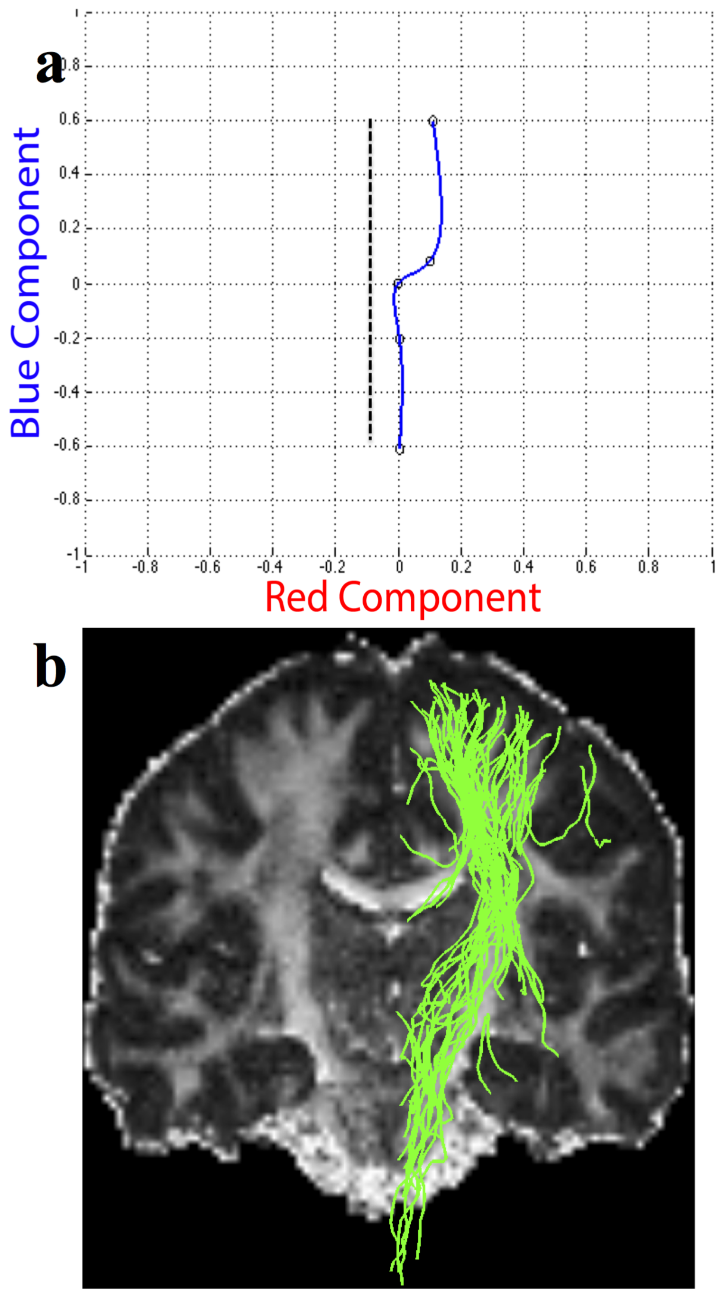


Figure 6-10: **a.** Approximate shape of Cortico-Spinal Tract (CST) and its axis of symmetry (black dotted line) drew by the operator; **b.** Extracted right CST fibers (green)

6.3.3 Integration of spatial information

In this section, we enhance the previous approach, and we present a methodology that integrates *QuickBundles* with a string-based fiber representation in such a way as to extract anatomically homogeneous WM fiber-bundles. *QuickBundles* (QB) [49] is, probably, the most famous approach that does not need a priori knowledge to isolate and extract WM fiber-bundles. Due to its simplicity, QB showed good results in terms of fiber-bundle extraction and execution time. However, as a side effect, the pure unsupervised approach used by it could lead to the extraction of anatomically incoherent regions. Indeed, the process of fiber generation adopted by QB does not take prior information from neuroanatomists into consideration. Nevertheless, this last information could play a key role for obtaining more satisfying results. As a consequence, this limitation could generate a bias in real applications, where anatomical information is important for analysis.

Given a set $F = \{f_1, f_2, \ldots, f_n\}$ of WM fibers to cluster and a set $M = \{m_1, m_2, \ldots, m_k\}$ of models, our approach consists of the following steps: *(i)* application of a string-based fiber representation formalism to construct the set T (resp., V) of the strings corresponding to F (resp., M); *(ii)* construction of a matrix D such that $D[i, j]$ indicates the dissimilarity degree between the string corresponding to f_i and the one associated with m_j; *(iii)* assignment of each fiber of F to at most one model of M on the basis of D in such a way as to produce a set $B = \{b_1, b_2, \ldots, b_k\}$ of WM fiber-bundles; interestingly, at this stage, it is not possible to distinguish symmetrical structures; *(iv)* application of QB to each bundle of B for overcoming this limitation. We conducted an experimental campaign to compare the performance of our approach with that of QB. As will be clear below, obtained results are very encouraging.

The following part is organized as follows. We first present QB in order to understand its functioning and we provide a technical description of the proposed approach. Then we illustrate the experimental campaign conducted to evaluate it. Finally, we draw our conclusion and delineate some possible future developments of this research.

QuickBundles QB [49] is an efficient unsupervised algorithm to cluster WM fiber-bundles. The idea behind it is simple. At each iteration, a given fiber of the tractography could be assigned to a pre-existing cluster or it could generate a new cluster. Initially, the first fiber is simply assigned to a first cluster containing only it. As for the other fibers, the assignment of a fiber to a cluster is performed according to a given threshold θ. If the distance between the current fiber and the centroid of at least one cluster is less than θ, the fiber is assigned to the cluster corresponding to the minimum distance. Otherwise, if there does not exist any cluster whose centroid has a distance from the current fiber less than θ, a new cluster is created and the fiber is assigned to it. This process is repeated until all the fibers in the tractography are assigned to a cluster. In order to measure the distance between two fibers, a new metric, called Minimum Average Direct Flip (MDF), is introduced. Differently from most classical clustering algorithms, like K-Means, in QB there is no re-assignment or updating step. So, when a fiber is assigned to one cluster, it is not possible for that fiber to change its cluster.

Technical description of the approach

Our approach joins together QB and MPED$_{SB}$ and aims at overcoming the main problem of the former by using the latter. Its first ingredient is a WM fiber-bundle reference model, which must represent an approximate shape of the fiber-bundle to extract. It could be obtained in two different ways, namely: *(i)* by exploiting a spline curve to draw the profile of the fiber-bundle of interest, or *(ii)* by importing the mean-line profile of the fiber-bundle of interest from an atlas of pre-labeled fiber-bundles. Both kinds of models can be constructed either by a generic user or with the support of an expert one. The second ingredient of our approach is a fiber representation formalism allowing fibers to be mapped on strings, which is the one presented and discussed in Section 6.3.2.

Once the two main ingredients of our approach have been defined, it is possible to describe it. Specifically, let $F = \{f_1, f_2, \ldots, f_n\}$ be a set of WM fibers to cluster and let $M = \{m_1, m_2, \ldots, m_k\}$ be the set of models. Our approach consists of the following steps:

- Construction of the set $T = \{t_1, t_2, \ldots, t_n\}$ of the strings corresponding to F and of the set $V = \{v_1, v_2, \ldots, v_k\}$ of the strings corresponding to M. For this purpose, the fiber representation formalism described above is applied.
- Construction of a $n \times k$ matrix D. The element $D[i, j]$ of D indicates the dissimilarity degree computed by applying MPED$_{SB}$ on the string t_i, associated with f_i, and the string v_j, associated with m_j.
- Assignment of each fiber of F to at most one model of M as follows:
  - for each row i of D, let μ be the minimum value of this row and let j_μ be the corresponding column;
  - if μ is lesser than a certain threshold Th then f_i is assigned to m_{j_μ}; otherwise, f_i is not assigned to any model.

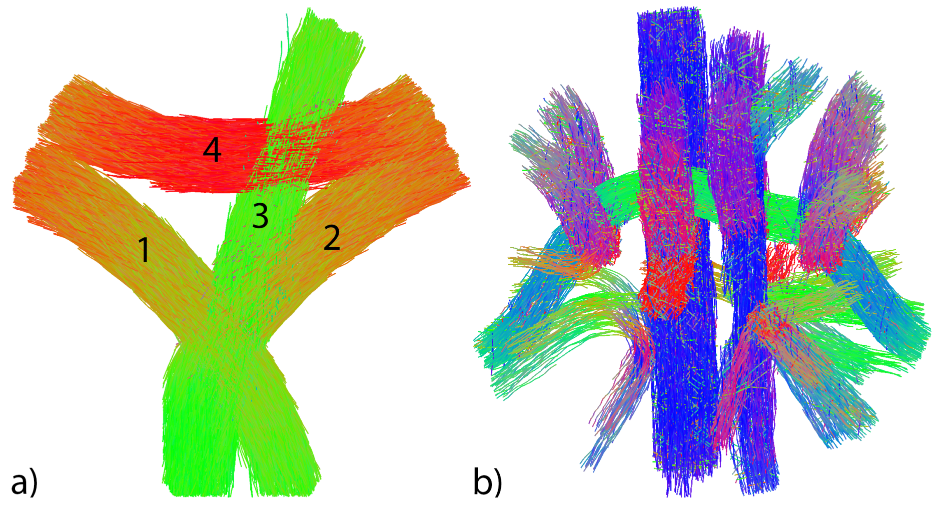


Figure 6-11: The two phantoms used in our experimental campaign

- At the end of this step, we have a set $B = \{b_1, b_2, \ldots, b_k\}$ of WM fiber-bundles, one for each model of M. However these bundles have a weak point. Indeed, the assignment approach above is incapable of distinguishing among symmetrical structures. To overcome this limitation, for each bundle $b_l \in B$, we apply QB to it. QB returns the same bundle b_l if it does not present a symmetrical structure. Otherwise, QB splits b_l into two symmetrical bundles b_l' and b_l''.

Experimental campaign

Our experimental campaign consisted of two phases. The former was devoted to tune our approach. The latter aimed to compare it with the classical QB. In both cases we exploited simulated diffusion phantoms, as well as some classical performance measures, namely, Precision, Recall, F-Measure and Overall.

Phase 1: Parameter Tuning. As for this phase, the input dataset consisted of the virtual phantom shown in Figure 6-11(a) and created by Phantomas [21]. In order to obtain the ground truth, experts segmented this phantom manually into 4 fiber-bundles, which are numbered in Figure 6-11(a). To find the best value of the input parameter Th, we considered different values of it ranging from 0.20 to 0.50. The corresponding values of Precision, Recall, F-Measure and Overall for the four models are reported in Figure 6-12. From the analysis of this figure, we can observe that, from $Th = 0.20$ to $Th = 0.36$, an increase of Th leads to an increase of at least one between Precision and

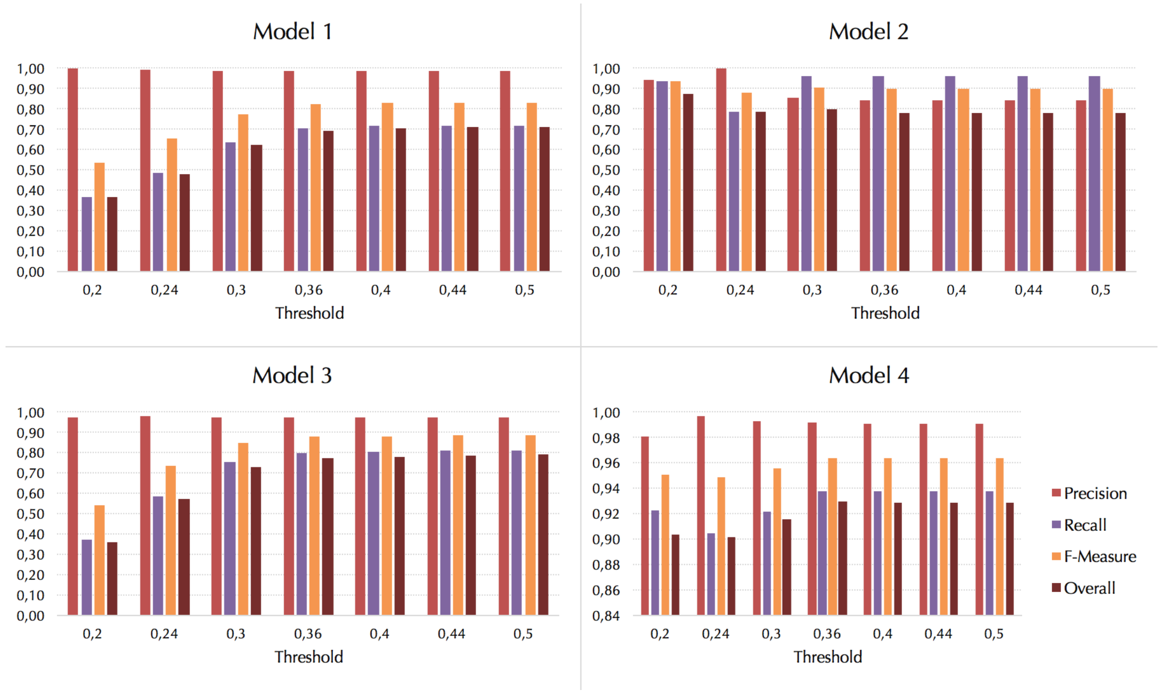


Figure 6-12: Variation of the four performance measures against the threshold Th for each model of the phantom of Figure 6-11(a)

Recall and to an increase of both F-Measure and Overall, which (we recall) are parameters combining Precision and Recall. Starting from $Th = 0.36$ no further increase of the values of performance measures can be observed in any model, and our approach shows a stable behavior. As a consequence, we chose to set Th to the middle of this range and we set it to 0.44.

Phase 2: Comparison with the classical QB. After having tuned Th, we applied the classical QB on the previous phantom to compare the performance of our approach (with $Th = 0.44$) with that of QB. The obtained results are reported in Table 6.9. From the analysis of this table we can observe that our approach shows a much higher average Precision, a slightly lower average Recall, a higher average F-Measure and a much higher average Overall than QB.

To obtain a (possible) confirmation of this result, we applied both our approach, with $Th = 0.44$, and QB on a second phantom shown in Figure 6-11(b). This consisted of a virtual diffusion MR phantom that accurately simulates the brain complexity with the fiber geometry used in the 2nd HARDI Reconstruction Challenge (ISBI 2013). Data were gen-

	Our approach				QB			
Model	**Precision**	**Recall**	**F-Measure**	**Overall**	**Precision**	**Recall**	**F-Measure**	**Overal**
1	0.99	0.72	0.83	0.71	0.96	1.00	0.98	0.95
2	0.84	0.96	0.90	0.78	0.54	0.77	0.64	0.12
3	0.97	0.81	0.88	0.78	0.54	0.77	0.64	0.12
4	0.99	0.94	0.96	0.93	1.00	0.93	0.96	0.93
Avg values	0.95	0.85	0.89	0.80	0.76	0.87	0.80	0.54

Table 6.9: Performance values obtained by our approach and QB when applied on the phantom of Figure 6-11(a)

	Our approach				QB			
Model	**Precision**	**Recall**	**F-Measure**	**Overall**	**Precision**	**Recall**	**F-Measure**	**Overal**
1	0.94	0.71	0.81	0.67	0.91	0.70	0.79	0.63
2	0.90	0.64	0.75	0.57	0.33	0.71	0.45	-0.76
3	0.78	0.41	0.54	0.29	0.31	0.75	0.44	-0.90
4	0.89	0.69	0.77	0.60	0.85	0.75	0.80	0.62
5	0.77	0.57	0.66	0.40	0.80	0.78	0.79	0.58
6	0.96	0.68	0.79	0.65	0.29	0.79	0.43	-1.13
7	0.94	0.29	0.45	0.27	0.14	0.82	0.24	-4.33
8	0.68	0.33	0.45	0.18	0.13	0.91	0.23	-4.96
Avg values	0.86	0.54	0.63	0.45	0.47	0.78	0.52	-1.28

Table 6.10: Performance values obtained by our approach and QB when applied on the phantom of Figure 6-11(b)

erated by means of Phantomas [21]. To obtain the ground truth, experts segmented all the fiber-bundles in the phantom manually. At the end of this task, they defined the approximate shapes of these fiber-bundles; in particular, they identified 17 models. Numbering these models in Figure 6-11(b) was not possible due to the 3D nature of this image.

Because of its fully unsupervised nature, QB was capable of extracting just 8 out of the 17 fiber-bundles of the phantom. Our approach, instead, extracted all the 17 fiber bundles. As a consequence, a comparison between our approach and QB was possible only for the 8 models detected by QB. Obtained results are reported in Table 6.10.

From the analysis of this table we can observe that QB generally shows a higher value of Recall than our approach (this can be observed in 7 out of 8 models), but lower values of Precision (which were much lower in 5 cases, lower in 2 cases and slightly higher in only 1 case). As for the combined parameters F-Measure and Overall, our approach shows better results than QB in 6 out of 8 cases. Finally, if we consider the average values of these measures, we obtain that our approach shows a better Precision, a better F-Measure, a better Overall and a worse Recall than QB.

As previously pointed out, this difference of behavior is due to the nature of the structures extracted by QB. In fact, the fiber-bundles obtained by QB are not isolated but they are merged with other structures that are spatially near to them (even if they present a completely different anatomical meaning). By contrast, the fiber-bundles obtained by our approach are “purer”, since they contain only anatomically uniform fibers, corresponding to the fiber bundles of our interest. In support of this reasoning, it is well known that, in this application field, Precision is much more important than Recall.

Conclusion

As a final consideration, we point out that our approach overcomes the main problem of QB, i.e., the possibility that it returns anatomically incoherent fiber clusters, in which the desired fiber-bundles are not isolated but merged with other structures spatially near to them (even if they present a completely different anatomical meaning). This work must not be considered as an ending point of our research efforts. Indeed, several developments are possible. First of all, we plan to extend our experiments from phantoms to real cases. After this, we would like to further improve QB in such a way as to correct fiber assignments to clusters when these assignments appear incorrect in a second time. Finally, we will work on the MPED$_{SB}$ constraint optimizations in such a way as to define an approach that allows the discovery of the best constrains for improving the extraction of specific WM fiber-bundles.

6.4 MPED for Electroencephalograms

In this section, we apply MPED, in conjunction with a network-based approach to help experts to investigate neurological disorders in which the connections among brain areas play a key role. Our approach uses a slightly different version of MPED, called *Consensus* MPED (cMPED) which operates as follows: it receives an electroencephalogram (EEG) of a patient, on which a preprocessing step for removing artifacts has been performed. Then, it constructs a network with nodes that correspond to electrodes and with edges that denote the disconnection level of the brain areas underlying the involved electrodes. The weight of an edge measures this disconnection level and is computed by applying the cMPED. Once this network has been constructed, some suitable projections can be derived from it, depending on the neurological disorder to investigate. After this, for each projection, a connection coefficient, taking both the number and the dimension of available cliques into account, is computed. The values of this coefficient for the various projections are employed to help experts in their analyses of neurological disorders. This section is organized as follows: in Section 6.4.1 we give some interesting and useful background in order to understand the problem. Section 6.4.2 presents the ingredients of our approach while Section 6.4.3 shows the approach itself. In Section 6.4.4 we describe its application by studying few neurological disorders and in Section 6.4.5 we compare our approach with another one recently proposed in the literature.

6.4.1 Background

In recent years, the incidence of neurological disorders is growing also because population is aging in most countries. At the same time, the efforts to design approaches capable of determining the onset of these disorders and of monitoring their course in the corresponding patients are intensifying [41, 62, 118]. Also the tools supporting neurologists in their activities are becoming more complex and sophisticated (think, for instance, of electroencephalograms with 256 electrodes, instead of the classical ones with 19 electrodes). The counterpart of these important advances is the need of handling huge amounts of data that experts have difficulty to analyze manually. In this scenario, automatic tools supporting experts to analyze available data are becoming mandatory.

Among the many diagnostic tools available to neurologists, EEG is one of the least invasive. For this reason, it is adopted to support the analyses of many neurological disorders. In the literature, many techniques to process EEG data have been proposed, and most of them are based on signal analysis [20, 66, 89, 106, 109, 115].

An EEG can be easily modeled as a network. Indeed, in the literature, several approaches that use networks to model EEGs and to investigate neurological disorders have been presented [36, 78, 81, 95, 105, 124]. After having modeled an EEG as a network, these approaches generally use basic concepts and metrics of network analysis (e.g., centrality measures, diameter, path length) to support an expert in her diagnosis.

It is well known that, in many neurological investigations, the key role is played by the connections between the brain areas. Network analysis provides some basic parameters to evaluate the connection level of a network. The most known of them are network density and clustering coefficient. However, these two parameters have not been specifically conceived for measuring the connection degree of a network. As a consequence, a challenging issue could be defining a parameter specifically thought for this purpose. Hopefully, this parameter could work better than density and clustering coefficient for evaluating the connection degree of a network. To define this parameter, we observe that cliques play a central role in identifying highly-connected portions of a network. Thus, they could represent the key concept in this task since the higher the number and the dimension of available cliques in a network and the higher the corresponding connection level.

However, a network associated with an EEG is totally connected, since a potential difference can be evaluated for each pair of its electrodes. As a consequence, we would have only one clique of maximum dimension in all cases. On the other side, potential difference between two electrodes is an indicator of the strength of the connection between them and, ultimately, between the corresponding brain areas. As a consequence, it is reasonable to use a metric derived from it to weigh the corresponding edges in the network. This metric could represent the distance, or the disconnection level, of the associated brain areas.

These edge weights could guide the analyses of the network and, ultimately, of the corresponding patient. As a matter of facts, it could be possible to define suitable projections of the network (i.e., suitable subnetworks) on the basis of them, and to evaluate the number and the dimension of the cliques on each of these projections. For instance, assume to obtain a derived network projection by removing the edges with the top $X\%$ of the weights. This is equivalent to not considering the edges with the highest distances or, in other words, to consider only the projected network embracing the most connected brain areas. On the basis of this reasoning, the number and the dimension of cliques in this projected network could surely play a prominent role in the definition of a connection coefficient for the whole original network.

In this scenario, a metric that, starting from the potential differences, can determine the disconnection level between two nodes is particularly important. In several contexts, related to sensors and biomedical engineering, string-based approaches, turning analog signals into discretized data streams, proved to be well suited. As a matter of facts, several approaches of this family turned out to be successful in analyzing, for instance, generic sensor network data [64, 127] and biomedical data streams [27, 99, 102]. In particular, once input data is transformed into a string, several metrics can be applied to measure the (dis)similarity between them. Some classical measures, like Hamming Distance, Longest Common Subsequence (LCS), and Levenshtein Distance [75], have been successfully adopted in this context. However, they are not well suited in some cases, particularly when the discretization step of similar signals may produce quite different discretized data, which makes it compulsory some form of approximation in string comparison.

6.4.2 Core ingredients

Consensus MPED

Here, we extend the problem of pairwise approximate string comparison to the analysis of sets of heterogeneous strings. This might be particularly important when we want to analyze the relative properties between pieces of information in a group, such as the connection level between groups of signals (see Section 6.4.3).

In order to carry out this task, we exploit our MPED, defining a slightly different version called *consensus* Multi-Parameterized Edit Distance (cMPED, for short), which is described next.

Intuitively, since strings to be examined are assumed to be correlated, the objective is both to identify hidden correlations and to compare strings on a common base. Then, given a set of strings and the set of matching schemas computed by MPED between them, we construct a consensus-based matching schema $\overline{M}$, common to all strings, and we compute the corresponding cMPED accordingly.

More formally, given a set $\mathcal{S} = \{s_1, \ldots, s_n\}$ of strings on a common alphabet Π obtained as the union of the involved alphabets, let $\mathcal{D}$ be the set of (MPED) distances between each pair s_i and s_j and let $\mathcal{M}$ be the set of the corresponding matching schemas. In particular, $d_{ij} \in \mathcal{D}$ represents the distance between s_i and s_j and $M_{ij} \in \mathcal{M}$ indicates the matching schema corresponding to d_{ij}, s_i and s_j.

A consensus matrix C can be built from $\mathcal{M}$ such that, for each pair of symbols $l \in \Pi$ and $m \in \Pi$:

$$C(l,m) = \frac{\sum_{i,j} M_{ij}(l,m)}{n(n-1)}.$$

Recall that $M_{ij}(l,m) = 1$ if symbols l and m match in the matching schema M_{ij}, and $M_{ij}(l,m) = 0$ otherwise. Since the sum is made over all the pairs of indices i, j corresponding to strings in $\mathcal{S}$, $C(l,m)$ measures the portion of matching schemas that agree on the match between symbols l and m, over all the $n(n-1)$ string pairs in $\mathcal{S}$.

In order to build $\overline{M}$, we consider the consensus stored in C among the pairs of symbols that should be considered to match. However, using a fixed threshold would be too weak, given the possibly high heterogeneity of situations coming from different distance sets. For instance, very dissimilar string sets will probably significantly disagree on many matchings of schemas, whereas very similar string sets will probably agree on most matchings.

To face this issue, given the maximum distance d_{max}, which can be in principle returned by MPED on $\mathcal{S}$, we introduce the following sigmoid-based threshold computation[3]:

$$Th(\mathcal{D}) = \frac{1}{1+e^{-3.5(min_{i,j}\{d_{ij}/d_{max}\}-0.95)}}$$

which represents the basis for the computation of each element of $\overline{M}$ as specified below:

$$\overline{M}(l,m) = \begin{cases} 1, & \text{if } C(l,m) \geq Th(\mathcal{D}) \\ 0, & \text{otherwise.} \end{cases}$$

The consensus-based matching schema $\overline{M}$ thus computed is then employed as the common matching schema between all string pairs in $\mathcal{S}$. Since it may differ from the original matching schemas that allowed the construction of $\mathcal{D}$, a new distance set $c\mathcal{D}$ of cMPEDs between strings in $\mathcal{S}$ is computed. This task is carried out by running MPED with the

[3]This formula has been empirically obtained over several sets of strings.

forced matching schema $\overline{M}$, which corresponds to computing the edit distance using $\overline{M}$ instead of symbol identity matching. Observe that $\overline{M}$ might not produce the lowest possible distance in some cases, and that it might not be compliant to parameters π_1, π_2 used in the previous step; however, this new matching schema expresses information shared among all considered string pairs and, consequently, it carries more contextualized and meaningful information.

Connection coefficient

As previously pointed out, in many application contexts (belonging to both the neurological scenarios and other ones), the key role is played by the (dis)connection degree of the network nodes. In the classical network analysis, several parameters to measure this feature have been proposed. The simplest one is network density, whereas the most common one is clustering coefficient.

Consider a network $\mathcal{N} = \langle V, E \rangle$, where V is the set of its nodes and E is the set of its edges. Each edge e_{ij} connects the nodes v_i and v_j. It can be represented as $e_{ij} = (v_i, v_j, w_{ij})$. Here, w_{ij} is a measure of "distance" between v_i and v_j and depends on the application context.

Network density indicates the portion of the potential connections in a network that are actual connections. If $\mathcal{N}$ is undirected, network density can be computed as:

$$\mathcal{D}_{\mathcal{N}} = \frac{|E|}{\frac{|V|(|V|-1)}{2}}$$

Clustering coefficient measures the degree to which nodes tend to cluster together. To define it, we must preliminarily introduce some concepts. In particular, a triad in $\mathcal{N}$ is a subnetwork consisting of three nodes. These nodes could be totally disconnected, partially connected or totally connected (in which case the triad is closed). The clustering coefficient $\mathcal{CC}_{\mathcal{N}}$ is defined as:

$$\mathcal{CC}_{\mathcal{N}} = \frac{\text{number of closed triads}}{\text{total number of triads}}$$

Clustering coefficient is already a good connectivity indicator. However, it is based on triads and, therefore, it focuses on the capability of nodes to be connected in small subnetworks (consisting all of three nodes). It could be extremely important to also evaluate the dimension of the totally connected subnetworks existing in a network. Clearly, this phenomenon is also partially measured by clustering coefficient, because the larger the totally connected subnetwork the higher the fraction of closed triads. However, in our opinion, clustering coefficient does not evidence this capability well and clearly. In fact, keeping $n+1$ nodes totally connected is exponentially more difficult than keeping n nodes totally connected, and this aspect is not considered by the clustering coefficient.

Just as the clustering coefficient is based on triads, so a new coefficient should be based on cliques. We recall that the concept of clique is an extension of the concept of triad. In fact, a clique of dimension n represents a subnetwork of n nodes totally connected such that it cannot be further extended to $n+1$ nodes. We call *connection coefficient* the new coefficient we are proposing in this section.

To define it, we must preliminarily introduce the set Cl of the cliques of $\mathcal{N}$ and the set $Cl_k \subseteq Cl$ of the cliques of dimension k belonging to $\mathcal{N}$. Starting from these definitions, the connection coefficient $\mathcal{C}_{\mathcal{N}}$ is defined as:

$$\mathcal{C}_{\mathcal{N}} = \sum_{k=1}^{|V|} |Cl_k| \cdot 2^k$$

where $|Cl_k|$ indicates the cardinality (i.e., the number of cliques) of Cl_k.

The definition of this coefficient is based on the following considerations: *(i)* both the dimension and the number of cliques are important as connectivity indicators; *(ii)* the concept of clique is intrinsically exponential; in other words, a clique of dimension $n+1$ is exponentially more complex than a clique of dimension n.

6.4.3 The proposed approach

After having examined the metrics employed in our approach, we are now able to describe it. Preliminarily, we point out that our approach could be applied to any context that can be modeled as a network such that: *(i)* each node has associated a string representing the features to investigate for the corresponding node in the context of interest; *(ii)* distances between strings weigh the strength of the relationships between the corresponding nodes and, therefore, between the corresponding entities; *(iii)* node connection plays a key role.

The investigation of several neurological disorders perfectly fits the scenario depicted above and, therefore, we specialize our approach to this context.

Having this in mind, consider a set $\mathcal{S}$ of EEGs, on which all the necessary pre-processing steps for eliminating the artifacts have been carried out.

Each EEG is then discretized to obtain strings from the analog data. It is well known that, whenever analog data is discretized, some amount of discretization error can always be present. A naïve discretization may affect the whole analysis process. Generally, the goal of a discretization step is to reduce this error as much as possible with respect to the modeling purposes. Several discretization techniques have been presented in the literature (see, e.g., [32, 70, 123]). In our approach, we employ the well known SAX (Symbolic Aggregate approXimation) algorithm [70]. Indeed, SAX allows: *(i)* a fast computation of discretized data, *(ii)* the dimensionality reduction of the analog signal into a string. SAX transforms an input data series with n measurement points into a string. It requires two key parameters, namely the length d of the output string, and the number of symbols to be considered for the string alphabet. SAX consists of two steps; first, it transforms input data into a Piecewise Aggregate Approximation (PAA) representation; then, it converts this last representation into a string. The adoption of PAA guarantees that symbols corresponding to input features have the same probabilities.

A network $\mathcal{N} = \langle V, E\rangle$ can be associated with each EEG of $\mathcal{S}$. Each node $v_i \in V$ corresponds to an electrode. Each edge $e_{ij} \in E$ connects the nodes v_i and v_j and can be represented as:

$$e_{ij} = (v_i, v_j, w_{ij})$$

Here, w_{ij} is a measure of the "distance" between v_i and v_j. It is an indicator of the disconnection level of v_i and v_j. In principle, as we pointed out above and as it will be clear in the following, the tasks composing our approach are orthogonal to the measure adopted for estimating w_{ij}. However, having in mind the considerations expressed in Section 6.4.2, we employed the average cMPED values computed throughout the whole EEG.

Observe that the network $\mathcal{N}$ presented above is totally connected. This makes it impossible its usage to investigate phenomena having the network connection level as a key factor, as the ones we want to analyze here. However, we argue that $\mathcal{N}$ is a weighted network, and the corresponding weights are indicators of the connection level of the corresponding nodes; as a consequence, they could be employed to discriminate strong connections from weak ones.

In order to discriminate node connections and, at the same time, to make the adoption of the connection coefficient possible, we need to construct a new model derived from $\mathcal{N}$. In performing this task, we implement a simple strategy, which allows us to obtain a more "user-friendly" and "expressive" model. In particular, we construct a new network, namely $\mathcal{N}_\pi$, by performing a "projection" of $\mathcal{N}$ with the objective of removing the "weakest" edges from it and by coloring the others on the basis of their weight. The details of the projection and coloring activities depend on the neurological disorder to analyze. To give an idea of them, in the following, we show a generic projection and coloring schema. In Section 6.4.4, we provide the detailed schema for CJD, epilepsy and AD. In our example schema, we use three colors, namely blue, red and green. Blue edges denote strong connections (i.e., small weights), red edges represent intermediate ones and, finally, green edges indicate weak connections. More formally, $\mathcal{N}_\pi$ can be represented as:

$$\mathcal{N}_\pi = \langle V, E_\pi \rangle$$

Here, the nodes of $\mathcal{N}_\pi$ are the same as the ones of $\mathcal{N}$. To define E_π, we employ the distribution of the weights of the edges of $\mathcal{N}$. Specifically, let max_E (resp., min_E) be the maximum (resp., minimum) weight of an edge of E. Starting from max_E and min_E, it is possible to define a parameter $step_E = \frac{max_E - min_E}{10}$, which represents the length of a "step" of the interval between min_E and max_E. We can define $d^k(E)$, $0 \leq k \leq 9$, as the number of the edges of E with weights belonging to the interval between $min_E + k \cdot step_E$ and $min_E + (k+1) \cdot step_E$. All these intervals are closed on the left and open on the right, except for the last one that is closed both on the left and on the right. E_π consists of all the edges of E belonging to $d^k(E)$, $k \leq th_{max}$. The edges of E_π are colored according to the scheme mentioned previously, i.e., $E_\pi = E_\pi^b \cup E_\pi^r \cup E_\pi^g$, where:

- $E_\pi^b = \left\{ e_{ij} \in E \mid e_{ij} \in \bigcup_{th_{min} \leq k \leq th_{br}} d^k(E) \right\}$,
- $E_\pi^r = \left\{ e_{ij} \in E \mid e_{ij} \in \bigcup_{th_{br} < k \leq th_{rg}} d^k(E) \right\}$,
- $E_\pi^g = \left\{ e_{ij} \in E \mid e_{ij} \in \bigcup_{th_{rg} < k \leq th_{max}} d^k(E) \right\}$.

Here, th_{min}, th_{br}, th_{rg} and th_{max} are suitable threshold values belonging to the integer interval $[0, 9]$. Their value depend on the neurological disorder to analyze and can be determined experimentally. In particular, as for the analysis of AD, we experimentally set $th_{min} = 0$, $th_{br} = 1$, $th_{rg} = 4$ and $th_{max} = 6$.

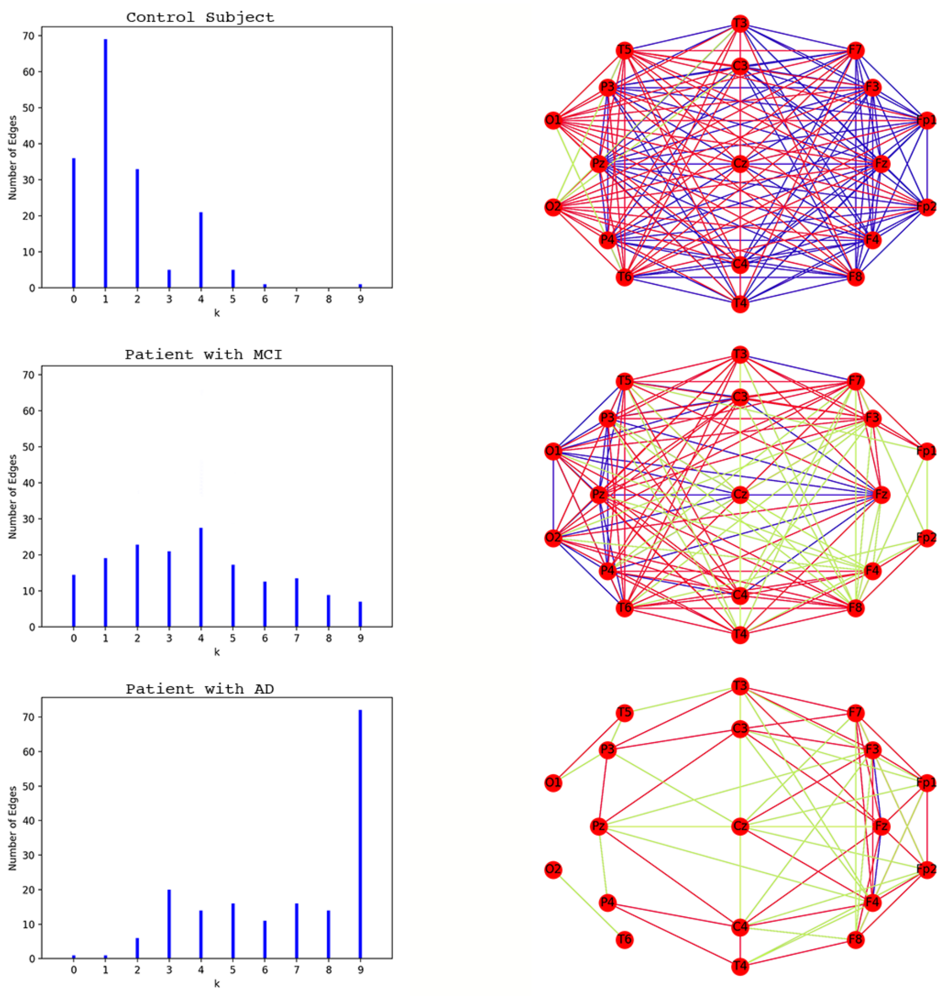


Figure 6-13: Distribution of the edge weights and colored network of a control subject, a patient with MCI and a patient with AD

To give an idea of the expressiveness of colored networks, in Figure 6-13 we report the distribution of the edge weights and the colored network of a control subject (resp., a patient with Mild Cognitive Impairment - hereafter, MCI - and a patient with AD). The disposal of nodes in the network reflects the 10-20 system, even if they are rotated 90 degrees clockwise. It is straightforward to observe that the control subject presents a weight distribution more biased on the left than the patient with MCI, who, in turn, presents a weight distribution more biased on the left than the patient with AD. A direct consequence of this fact is that the colored network of the patient with AD presents lesser and weaker edges than the colored network of the patient with MCI that, in turn, presents lesser and weaker edges than the colored network of the control subject.

In order to quantify this phenomenon, in Table 6.11, we report the values of some measures characterizing the three colored networks shown in Figure 6-13. Specifically, the considered measures are: *(i)* the total number of colored edges; *(ii)* the total number of blue (resp., red, green) edges[4]; *(iii)* the percentage of colored edges against the total number of original edges; *(iv)* the percentage of blue (resp., red, green) edges against the total number of original edges. The quantitative values reported in Table 6.11 fully confirm the qualitative analysis mentioned above.

Parameter	*Control Subject*	*Patient with MCI*	*Patient with AD*
Total number of colored edges	170	141	69
Total number of blue edges	105	35	2
Total number of red edges	59	75	40
Total number of green edges	6	31	27
Percentage of colored edges	99.4%	82.5%	40.3%
Percentage of blue edges	61.4%	20.5%	1.2%
Percentage of red edges	34.5%	43.8%	23.4%
Percentage of green edges	3.5%	18.1%	15.8%

Table 6.11: Quantitative measures representing the colored networks of Figure 6-13

From the previous description, it emerges that, thanks to $\mathcal{N}_\pi$, we reached our objectives of: *(i)* discriminating strong connections from weak ones; *(ii)* representing connections in a friendly and expressive way, and *(iii)* making the adoption of the connection coefficient possible. This last objective is extremely important because, as it will be clear in the following, the connection level is a very important indicator in the analysis of several neurological disorders.

We are now able to provide a general schematization of our approach. Specifically, it consists of the following steps:

1. Construct the set $\mathcal{S}$ of the EEGs of interest by performing electroencephalograms of the subjects to analyze.
2. Preprocess the EEGs of $\mathcal{S}$ to remove artifacts.
3. For each preprocessed EEG of $\mathcal{S}$:
    - 3.1 Compute *c*MPED for each pair of electrodes.
    - 3.2 Compute the corresponding network $\mathcal{N}$ using *c*MPED values as edge weights.
    - 3.3 Derive the "projected" network(s) from $\mathcal{N}$.
    - 3.4 Compute the connection coefficient for the "projected" network(s).
    - 3.5 Analyze the connection coefficients to draw the suitable conclusions.

[4]Recall that blue edges are the strongest ones, red edges have an intermediate weight, whereas green edges are the weakest ones.

As for Step 3.5, once the projected networks, and the corresponding connection coefficient, have been obtained for all the EEGs of $\mathcal{S}$, many analyses can be performed on them to extract useful information. These analyses depend on the neurological disorder the expert is investigating. To give an idea of them, in the next section, we consider three neurological disorders, namely CJD, epilepsy and AD.

6.4.4 Investigating neurological disorders

In this section, we provide an overview of how the general approach presented in Section 6.4.4 can be specialized in the investigation of three neurological disorders, namely CJD, epilepsy and AD.

Creutzfeldt-Jacob Disease

CJD is a rapidly progressive, uniformly fatal Transmissible Spongiform Encephalopathy (TSE), characterized by the accumulation of a variant of the host encoded cellular prion protein in the brain [118, 119]. Sporadic CJD (hereafter, sCJD) represents the most common form of CJD; in fact, it occurs worldwide in 84% of cases of CJD. It has an annual mortality rate of 1.39 per million. An early and reliable diagnosis of CJD is extremely important to exclude other, potentially treatable, causes of encephalopathies. However, the early diagnosis of this disease is complicated by the extreme heterogeneity of its clinical presentation. EEG has always been, and still is, one of the main methods to perform clinical diagnosis of CJD. In fact, in the EEG of patients with sCJD, it is often possible to observe three-phase periodic spikes with sharp waves known as “Periodic Sharp Wave Complexes” (hereafter, PSWCs). More specifically, PSWCs were reported to occur in the EEG tracings of about two-thirds of patients with sCJD. For this reason, they were included in the World Health Organization diagnostic classification criteria of sCJD [117–119]. In the past, approaches to investigating PSWCs in the EEGs of patients with sCJD were mainly based on signal processing [109, 115]. By contrast, to the best of our knowledge, no network analysis based approach to investigating the CJD phenomenon has been previously proposed in the literature. In sCJD the investigation of the connection degree of the brain areas is extremely important. In particular, in the past literature, it was shown that brain areas are more connected in presence of PSWCs than in absence of them [112]. This implies that our approach could really represent a useful tool for investigating PSWCs.

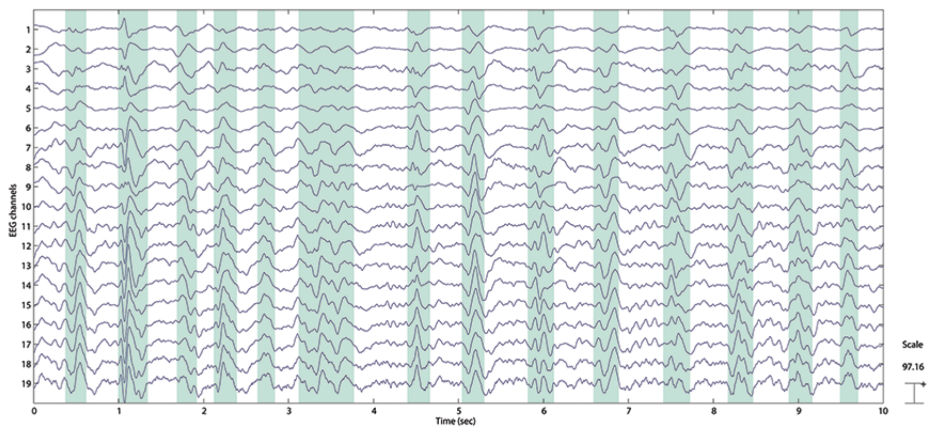


Figure 6-14: Partitioning of an EEG into segments with PSWCs and without PSWCs

To give an idea of how our approach can be applied in this case, consider an EEG of a patient with sCJD. This EEG can be segmented in such a way as to separate the tracing segments with PSWCs from those without PSWCs (Figure 6-14). As a consequence of this task, several tracing segments can be found in the EEG, which could be grouped in two distinct sets, namely those containing PSWCs and those not containing PSWCs.

At this point, a network $\mathcal{N}$ (resp., $\overline{\mathcal{N}}$) representing the set of the EEG segments with PSWCs (resp., without PSWCs) can be defined. Specifically:

$$\mathcal{N} = \langle V, E \rangle \qquad \qquad \overline{\mathcal{N}} = \left\langle V, \overline{E} \right\rangle$$

Here, V is the set of the nodes of $\mathcal{N}$ and $\overline{\mathcal{N}}$. Each node $v_i \in V$ corresponds to an electrode. E (resp., $\overline{E}$) is the set of the edges of $\mathcal{N}$ (resp., $\overline{\mathcal{N}}$). Each edge $e_{ij} \in E$ (resp., $\overline{e_{ij}} \in \overline{E}$) connects the nodes v_i and v_j. It can be represented as $e_{ij} = (v_i, v_j, w_{ij})$ (resp., $\overline{e_{ij}} = (v_i, v_j, \overline{w_{ij}})$). w_{ij} (resp., $\overline{w_{ij}}$) denotes the values of the cMPED between v_i and v_j, averaged for all segments with (resp., without) PSWCs, and, as usual, it is an indicator of the disconnection level of v_i and v_j. Once $\mathcal{N}$ and $\overline{\mathcal{N}}$ have been constructed, it is possible to derive $\mathcal{N}_\pi$ and $\overline{\mathcal{N}_\pi}$.

In Figure 6-15, we report the colored networks $\mathcal{N}_\pi$ and $\overline{\mathcal{N}_\pi}$ for a patient with sCJD. The disposal of the nodes in the networks reflects the 10-20 system, even if they are rotated 90 degrees clockwise.

Observe that, in this figure, the number of edges of $\mathcal{N}_\pi$ is higher than the one of $\overline{\mathcal{N}_\pi}$. Furthermore, in $\mathcal{N}_\pi$, there are much more blue and red edges than those in $\overline{\mathcal{N}_\pi}$, where,

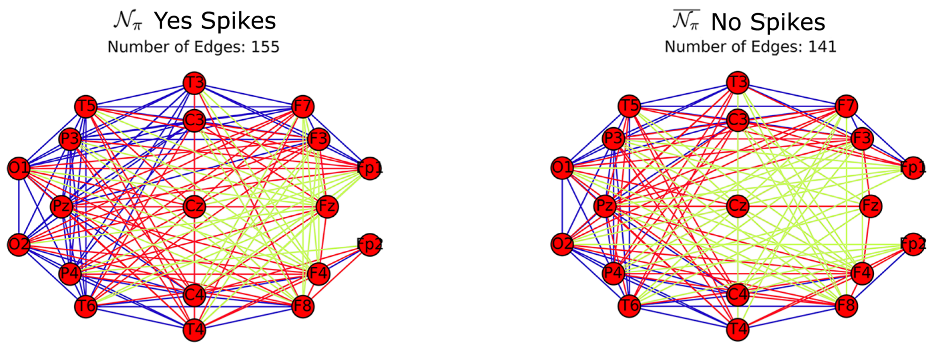


Figure 6-15: Colored Networks $\mathcal{N}_\pi$ and $\overline{\mathcal{N}_\pi}$ for a patient with CJD

instead, a higher number of green edges can be observed. When passing from $\mathcal{N}_\pi$ to $\overline{\mathcal{N}_\pi}$ the number of edges decreases of $\frac{|\overline{E}|-|E|}{\overline{E}}$ = -9.03%.

By computing the corresponding connection coefficient, we obtain:

$$\mathcal{C}_{\mathcal{N}_\pi} = 196642 \qquad \mathcal{C}_{\overline{\mathcal{N}_\pi}} = 51704 \qquad \frac{\mathcal{C}_{\overline{\mathcal{N}_\pi}}-\mathcal{C}_{\mathcal{N}_\pi}}{\mathcal{C}_{\overline{\mathcal{N}_\pi}}} = -73.71\%$$

This example shows that our approach is really useful to support experts in their analysis of patients with sCJD. In fact, in patients with PSWCs, the connection coefficient is much higher than in control subjects. Interestingly, in this case, the decreases of density and clustering coefficients when passing from $\mathcal{N}_\pi$ to $\overline{\mathcal{N}_\pi}$ are as follows:

$$\frac{\mathcal{D}_{\overline{\mathcal{N}_\pi}}-\mathcal{D}_{\mathcal{N}_\pi}}{\mathcal{D}_{\overline{\mathcal{N}_\pi}}} = -9.03\% \qquad \frac{\mathcal{CC}_{\overline{\mathcal{N}_\pi}}-\mathcal{CC}_{\mathcal{N}_\pi}}{\mathcal{CC}_{\overline{\mathcal{N}_\pi}}} = -3.46\%$$

These values clearly evidence that our connection coefficient is really more powerful in discriminating patients with PSWCs from patients without PSWCs.

Clearly, this study could stimulate neurologists to organize medical campaigns aimed to verify *how much* our approach is capable of helping experts in the analysis of patients with sCJD. For instance, several sets of patients with sCJD could be selected and parameters as sensitivity, specificity and precision could be computed.

Epilepsy

Epilepsy is a neurological disorder characterized by recurrent seizures. These are still considered unpredictable, despite the huge efforts spent in recent years by the scientific community to develop predictive algorithms. One of the main tools adopted to perform the analysis of epileptic patients is EEG. To facilitate the diagnosis of this disorder, worldwide researchers are working to automatically mark the critical events occurring in an EEG, as well as to extract meaningful features from EEG signals. A specific form of epilepsy is Childhood Absence Epilepsy (CAE), an idiopathic generalized epileptic disorder [40, 41] characterized by recurrent "absence seizures" that cause disruption of awareness and are often associated with staring. Detecting ictal states in patients with CAE is a very delicate and time consuming task for a neurologist, who has to analyze the whole EEG. Our approach could help her in this task.

To give an idea of how our approach can be applied in this case, consider an EEG of a patient with CAE. Starting from it, a network $\mathcal{N}$ can be constructed, as specified in Section 6.4.4. After this, two network projections, namely *black* and *white* ones, can be constructed. Formally speaking, these two network projections can be defined as $\mathcal{N}_\pi^{blk} = \langle V, E_\pi^{blk} \rangle$ and $\mathcal{N}_\pi^{wht} = \langle V, E_\pi^{wht} \rangle$. Here:

- $E_\pi^{blk} = \left\{ e_{ij} \in E \mid e_{ij} \in \bigcup_{0 \leq k \leq 4} d^k(E) \right\}$,
- $E_\pi^{wht} = \left\{ e_{ij} \in E \mid e_{ij} \in \bigcup_{5 \leq k \leq 9} d^k(E) \right\}$.

We observe that $\mathcal{N}^{wht}$ considers the five intervals of edge distribution characterized by the heaviest weights, whereas $\mathcal{N}^{blk}$ encompasses the other ones. Now, it could be possible to compute, on a time slot base, the value of the connection coefficients $\mathcal{C}_{\mathcal{N}_\pi^{blk}}$ and $\mathcal{C}_{\mathcal{N}_\pi^{wht}}$ of $\mathcal{N}_\pi^{blk}$ and $\mathcal{N}_\pi^{wht}$, respectively.

Since edge weights represent distances, on the basis of the results of [95], we can expect that, in presence of an ictal state, $C_{\mathcal{N}_\pi^{blk}}$ presents a maximum, whereas $C_{\mathcal{N}_\pi^{wht}}$ shows a minimum. This is explained by the fact that, during ictal states, the weights of the edges tend to decrease and, therefore, several edges disappear from $\mathcal{N}^{wht}$ and appear in $\mathcal{N}^{blk}$.

In Table 6.12, we report data about figures specified by an expert neurologist when she examined a whole EEG of a patient with CAE. The physician identified 8 seizures, which took place into the time-slots specified in this table. We use this table as a starting point and a benchmark of accuracy for the detection of ictal states performed by our approach.

In Figures 6-16 and 6-17, we plot the values of the connection coefficient (y axis) for each time-slots (x axis) for the EEG into consideration and for $\mathcal{N}_\pi^{blk}$ and $\mathcal{N}_\pi^{wht}$, respectively. Clearly, in these figures, it is straightforward to observe that there are some time-slots in which connection coefficient is several orders of magnitude lower than others. The important result is that those time-slots are exactly the ones that the neurologist spotted as ictal

Seizure id	*Starting time-slot*	*Ending time-slot*
1	4	26
2	120	122
3	165	205
4	306	332
5	449	451
6	470	496
7	642	659
8	891	913

Table 6.12: Table produced by a neurologist regarding starting and ending time-slots for each seizure of the EEG into consideration

states. Thus, without having to manually analyze the whole EEG of a patient, thanks to our approach, ictal states can be easily distinguished from the others.

Again, this study is simply a starting point. In fact, in the future, we plan to perform research efforts to try to make our approach capable of predicting the next seizures of a patient. The ultimate goal is, again, stimulating neurologists to organize medical campaigns for verifying how much our approach is capable of helping experts in the analysis of patients with CAE and, more in general, with several forms of epilepsy.

In Figures 6-18 and 6-19, we plot the values of the density coefficient $\mathcal{D}_{\mathcal{N}_\pi^{blk}}$ and $\mathcal{D}_{\mathcal{N}_\pi^{wht}}$ for the same EEG of Figures 6-16 and 6-17. It is straightforward to observe that density coefficient is less capable of clearly distinguishing ictal states from the other ones. As a matter of fact, if some form of evidence of ictal states can be observed for $\mathcal{D}_{\mathcal{N}_\pi^{wht}}$, for $\mathcal{D}_{\mathcal{N}_\pi^{blk}}$ this evidence disappears.

Analogous conclusions can be drawn for clustering coefficient, with values for $\mathcal{CC}_{\mathcal{N}_\pi^{blk}}$ and $\mathcal{CC}_{\mathcal{N}_\pi^{wht}}$ for the same EEG shown in Figures 6-16 and 6-17 that are reported in Figures 6-20 and 6-21.

Alzheimer's Disease

In recent years, the efforts to design approaches capable of determining the onset of AD in advance are intensifying [58, 97]. Even if this issue is challenging, it is extremely complex. A further important issue that makes the diagnosis on these patients difficult concerns the fact that they, due to the very nature of their disease, do not easily undergo examinations, like Magnetic Resonance Imaging, which force them to stay motionless for a long time. A non-invasive and well tolerated examination, which can be done on patients with AD, is EEG. The possibility of using EEG for characterizing patients with AD is evidenced in [62, 66].

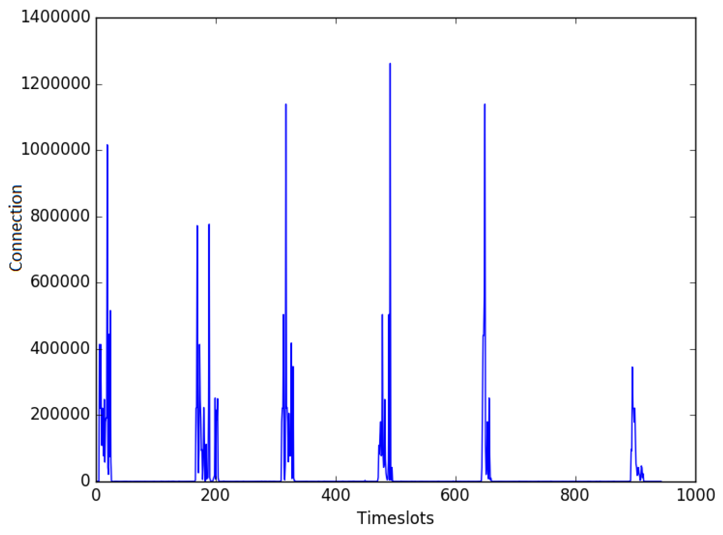


Figure 6-16: Connection coefficient for the network $\mathcal{N}^{blk}$

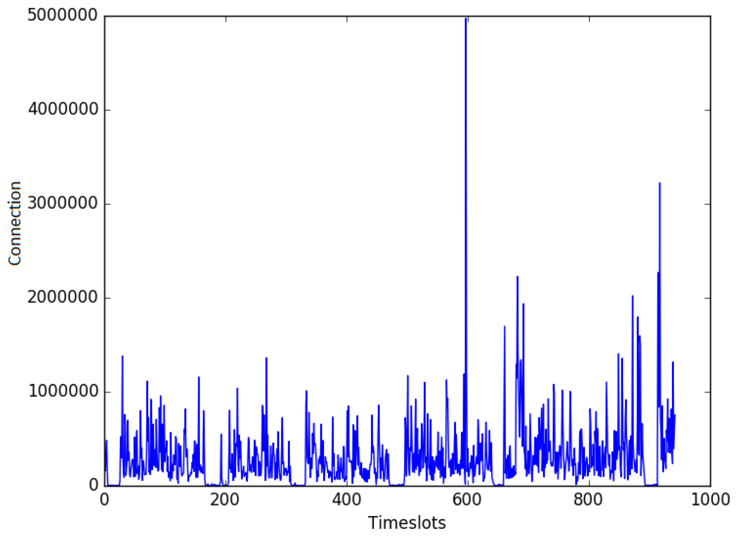


Figure 6-17: Connection coefficient for the network $\mathcal{N}^{wht}$

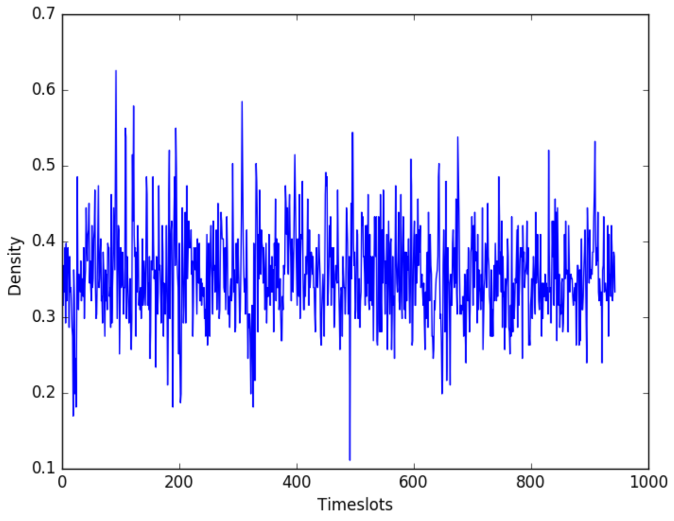


Figure 6-18: Density for the network $\mathcal{N}^{blk}$

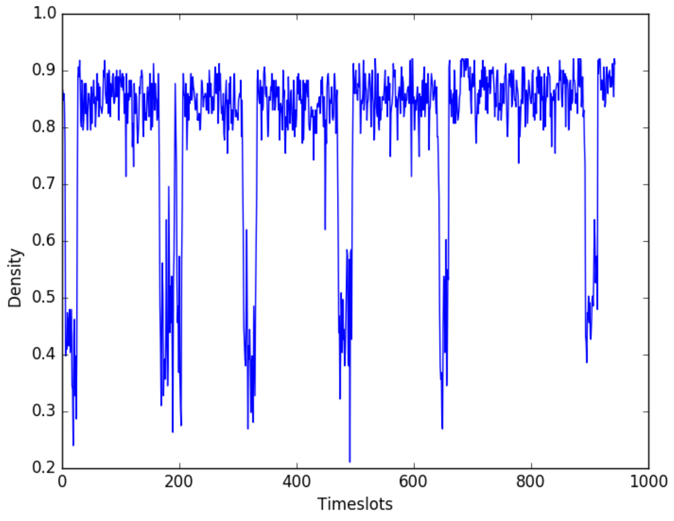


Figure 6-19: Density for the network $\mathcal{N}^{wht}$

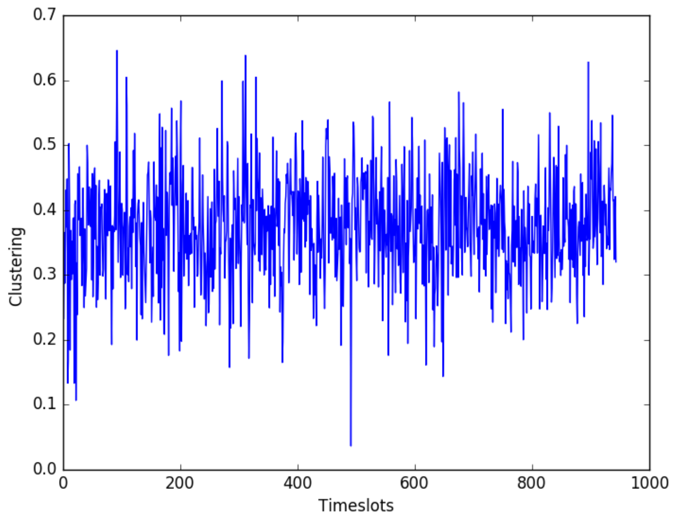


Figure 6-20: Clustering coefficient for the network $\mathcal{N}^{blk}$

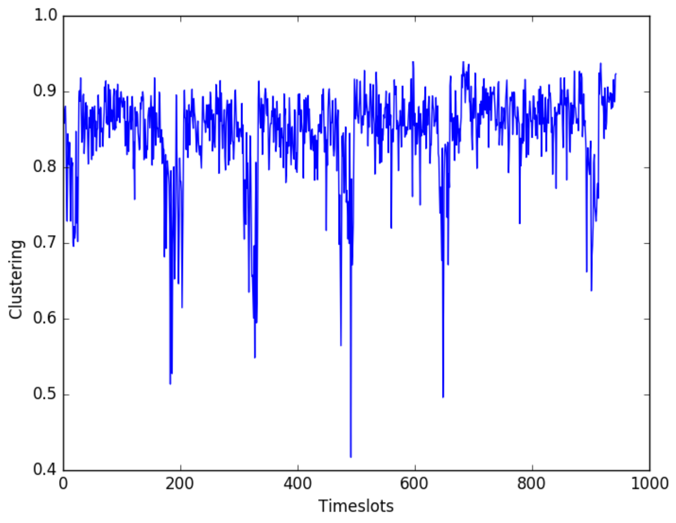


Figure 6-21: Clustering coefficient for the network $\mathcal{N}^{wht}$

In fact, the EEGs of patients with AD present some peculiarities, namely slowing, reduced complexity and perturbations in synchrony. As specified in [93], MCI can be prodromal for AD. In fact, several papers suggest that patients with MCI tend to convert to AD with a rate of about 10-15% annually [34]. For this reason, a large variety of approaches aiming at characterizing both MCI and AD have been proposed in the past literature. Several of these approaches are based on the analysis of EEG. In fact, it is well known that, in AD progression and in MCI progression towards AD, a key role is played by the loss of connectivity among the different cortical areas.

In this context, our approach can help the experts in evaluating whether a certain patient is presumably suffering from MCI or from AD[5] and, even more interesting, in evaluating if, with the passing of time, an individual with MCI is converting to AD or not.

To give an idea of how our approach can be applied in this case, consider the EEG of a patient p performed at two different time instants, namely t_0 and t_1, being t_1 some months after t_0. Starting from them, two networks $\mathcal{N}_{0_\pi} = \langle V, E_{0_\pi} \rangle$ and $\mathcal{N}_{1_\pi} = \langle V, E_{1_\pi} \rangle$ can be constructed.

Here: $E_{0_\pi} = E^b_{0_\pi} \cup E^r_{0_\pi} \cup E^g_{0_\pi}$; $E^b_{0_\pi}$, $E^r_{0_\pi}$, and $E^g_{0_\pi}$ are computed by applying the definitions of Section 6.4.4.

$E_{1_\pi} = E^b_{1_\pi} \cup E^r_{1_\pi} \cup E^g_{1_\pi}$; $E^b_{1_\pi}$, $E^r_{1_\pi}$, and $E^g_{1_\pi}$ are also computed as specified in Section 6.4.4. However, in their computation, the reference interval $[min_E, max_E]$ for the edge weight distribution is the one of the edges of E_{0_π} and not the one of the edges of E_{1_π}. As will be clear below, this is necessary for making it possible the comparison of the course of the neurological situation of p when passing from t_0 to t_1.

In order to give an idea of the capabilities of our approach in this application context, in Figure 6-22, we illustrate the networks $\mathcal{N}_{0_\pi}$ and $\mathcal{N}_{1_\pi}$ for two patients with MCI at t_0, who remained with MCI at t_1. Both the images and the corresponding quantitative descriptions, reported in Table 6.13, show that there is no substantial change in the networks of these patients when passing from t_0 to t_1.

A completely different situation can be observed in Figure 6-23, where we show the networks $\mathcal{N}_{0_\pi}$ and $\mathcal{N}_{1_\pi}$ for two patients with MCI at t_0, who converted to AD at t_1. In this case, both the images and the corresponding quantitative descriptors, reported in Table

[5]Recall that a clinical diagnosis of AD is very difficult, especially at the early stages of the disease; for a definitive diagnosis of AD, the biopsy of brain tissues is even necessary.

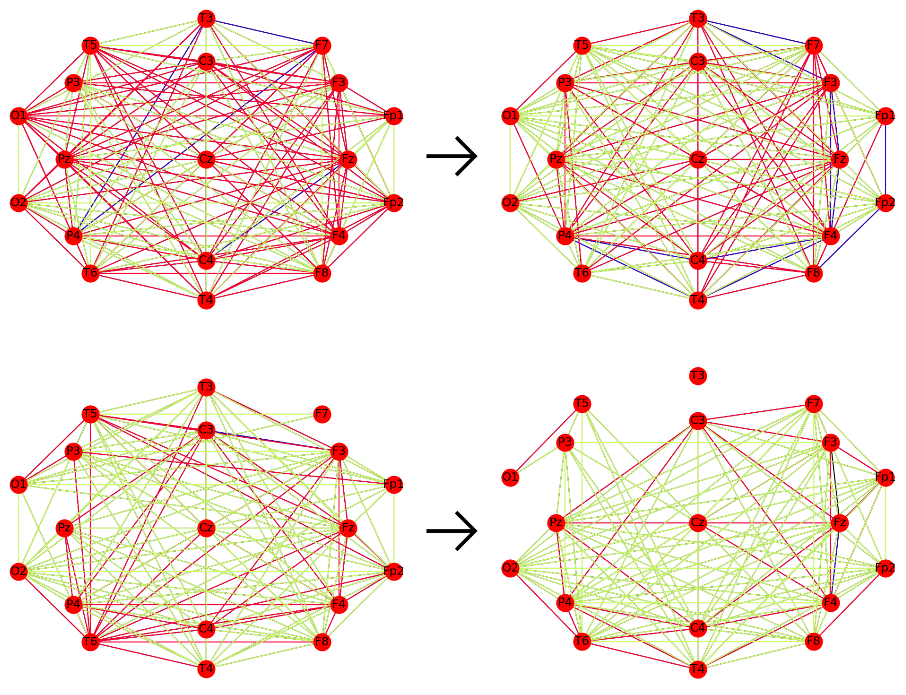


Figure 6-22: The networks $\mathcal{N}_{0_\pi}$ and $\mathcal{N}_{1_\pi}$ for two patients with MCI at both t_0 and t_1

6.14, evidence a substantial change in the networks of these patients when passing from t_0 to t_1.

Let us now examine connection coefficient. In particular, let us consider $\frac{\mathcal{C}_{\mathcal{N}_{1_\pi}} - \mathcal{C}_{\mathcal{N}_{0_\pi}}}{\mathcal{C}_{\mathcal{N}_{0_\pi}}}$, i.e., the variation of the connection coefficient when passing from t_0 to t_1. For the two patients MCI-MCI of Figure 6-22, this variation is equal to -25.00% and -4.96%, whereas for the two patients MCI-AD of Figure 6-23, this variation is equal to -89.06% and -99.41% These results show that our approach can really help experts to evaluate the conversion of a patient from MCI to AD.

Let us consider, now, the variation of network density $\frac{\mathcal{D}_{\mathcal{N}_{1_\pi}} - \mathcal{D}_{\mathcal{N}_{0_\pi}}}{\mathcal{D}_{\mathcal{N}_{0_\pi}}}$ for the same patients. We obtain that it is equal to -2.51% and -11.19% for the two MCI-MCI patients and is equal to -3.29% and -71.94% for the two MCI-AD patients. An analogous task performed on clustering coefficient returns that $\frac{\mathcal{CC}_{\mathcal{N}_{1_\pi}} - \mathcal{CC}_{\mathcal{N}_{0_\pi}}}{\mathcal{CC}_{\mathcal{N}_{0_\pi}}}$ is equal to -1.61% and -3.19% for the two MCI-MCI patients and is equal to -5.48% and -39.89% for the two MCI-AD patients.

These results show that, also for this scenario, the connection coefficient introduced is better than the classical network density and clustering coefficient parameters in discriminating different neurological situations.

Parameter	$\mathcal{N}_{0_\pi}$ (I patient MCI-MCI)	$\mathcal{N}_{1_\pi}$ (I patient MCI-MCI)	$\mathcal{N}_{0_\pi}$ (II patient MCI-MCI)	$\mathcal{N}_{0_\pi}$ (II patient MCI-MCI)
Total number of colored edges	159	155	134	118
Total number of blue edges	7	12	2	2
Total number of red edges	98	65	49	31
Total number of green edges	54	78	83	85
Percentage of colored edges	92.98%	90.64%	78.36%	69.00%
Percentage of blue edges	4.09%	7.02%	1.17%	1.17%
Percentage of red edges	57.30%	38.01%	28.65%	18.13%
Percentage of green edges	31.57%	45.61%	48.54%	49.71%

Table 6.13: Quantitative measures representing the networks $\mathcal{N}_{0_\pi}$ and $\mathcal{N}_{1_\pi}$ for the two patients considered in Figure 6-22

Parameter	$\mathcal{N}_{0_\pi}$ (I patient MCI-AD)	$\mathcal{N}_{1_\pi}$ (I patient MCI-AD)	$\mathcal{N}_{0_\pi}$ (II patient MCI-AD)	$\mathcal{N}_{0_\pi}$ (II patient MCI-AD)
Total number of colored edges	152	147	139	39
Total number of blue edges	39	5	33	1
Total number of red edges	84	52	60	8
Total number of green edges	29	90	46	30
Percentage of colored edges	88.89%	85.96%	81.28%	22.80%
Percentage of blue edges	22.81%	2.92%	19.29%	0.58%
Percentage of red edges	49.12%	30.40%	35.08%	4.68%
Percentage of green edges	16.95%	52.63%	26.90%	17.54%

Table 6.14: Quantitative measures representing the networks $\mathcal{N}_{0_\pi}$ and $\mathcal{N}_{1_\pi}$ for the two patients considered in Figure 6-23

6.4.5 Discussion

In the previous section, we have shown how our approach can be useful to support experts in their analyses of some neurological disorders. We have also compared our connection coefficient with two classical network analysis parameters and we have seen that it is better than them in discriminating neurological situations in which the connection level of brain areas plays a key role.

However, it is extremely interesting to evaluate our approach as a whole in at least one scenario. One way to do this task is to choose a scenario among the three ones mentioned above and to verify the behavior of both our approach and another one that proved to be well suited therein.

To carry out this task, we chose the scenario described in Section 6.4.4 and the approach based on Permutation Disalignment Index (hereafter, PDI) presented in [80]. Indeed, PDI proved to be well suited in quantifying the overall coupling strength between EEG signals associated with MCI progression towards AD. Furthermore, in this scenario, PDI was shown to outperform both Coherence and Dissimilarity Index. The former is a well known

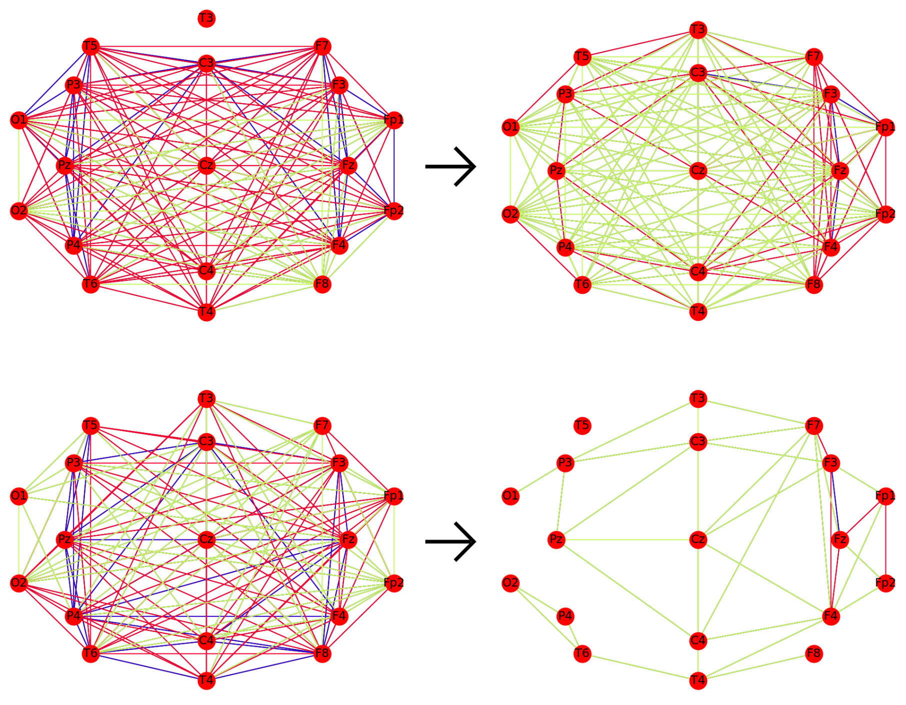


Figure 6-23: The networks $\mathcal{N}_{0_\pi}$ and $\mathcal{N}_{1_\pi}$ for two patients with MCI at t_0, who converted to AD at t_1

parameter adopted in neurological research; the latter is a nonlinear and symbolic measure that proved to be promising in the pairwise analysis of EEG data. In [80], the authors use boxplots and PDI to verify whether a patient with MCI at t_0 converts to AD at t_1 or not.

Before proceeding with the description of this activity, we point out that, here, we do not aim at comparing the two approaches for establishing if one of them is better than the other. In fact, for this last purpose, it would be necessary to perform a theoretical investigation and an experimental campaign totally centered on AD, as well as to analyze many patients according to the protocols typical of medical experimental campaigns in this research area. Actually, this is not the purpose of this work. Indeed, we aim at showing that our approach can be useful to help experts in their analyses of those neurological disorders strongly characterized by the decrease of the connections between brain areas. This way, we hope to stimulate neurologists to organize medical campaigns for quantifying how much our approach can help experts in the analysis of patients with specific neurological diseases. In particular, we apply the approach of [80] to the same four patients considered in Section 6.4.4 to verify whether the results returned by our approach are in line with the ones returned by the approach of [80].

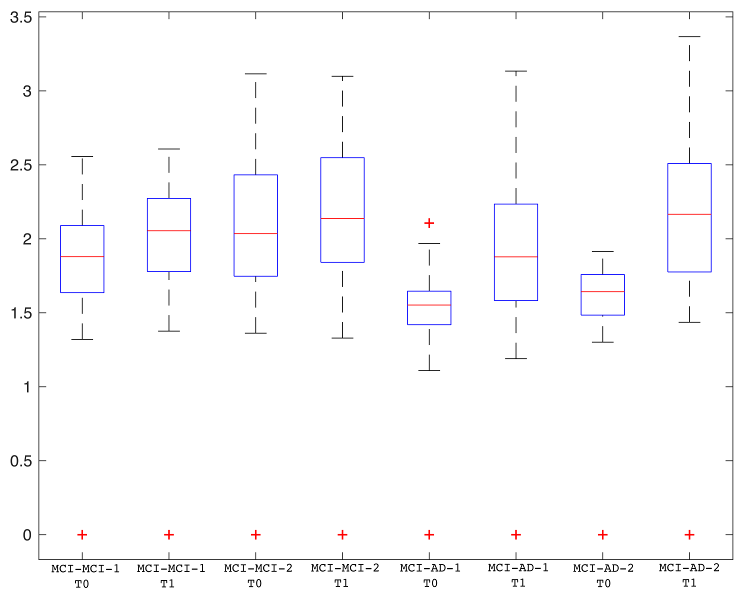


Figure 6-24: Results of the application of the approach of [80] to the patients of Figures 6-22 and 6-23

In Figure 6-24, we report the boxplots of the four patients. In Table 6.15, we present the values of some parameters helping to quantify the results shown in Figure 6-24. From the analysis of Figure 6-24 and Table 6.15, and from the comparison of the results represented therein with the ones of Section 6.4.4 (summarized in Table 6.16), obtained by applying our approach to the same patients, we can see that our approach returns results in line with (and even better than) the ones returned by the approach of [80]. Indeed, in these (limited number of) cases, it proved to be more capable of discriminating the conversion form MCI to AD than the approach of [80].

	Variation of medians from $T0$ to $T1$	Variation of 25th percentile from $T0$ to $T1$	Variation of 75th percentile from $T0$ to $T1$
MCI-MCI-1	9.04%	8.59%	9.13%
MCI-MCI-2	4.93%	5.75%	4.53%
MCI-AD-1	20.65%	11.27%	35.97%
MCI-AD-2	31.70%	19.59%	43.43%

Table 6.15: Quantitative measures representing the results shown in Figure 6-24

This result is extremely encouraging and, in our opinion, it allows us to say that our approach reached its goal to help neurologists in their analyses of some disorders.

In Section 6.4.3, presenting the proposed approach, we observed that, in principle, the tasks composing our approach are orthogonal to the measure adopted for estimating w_{ij}.

	Variation of $\mathcal{C}_{\mathcal{N}}$ from $T0$ to $T1$
$P_{MCI-MCI-1}$	-25.00%
$P_{MCI-MCI-2}$	-4.96%
$P_{MCI-AD-1}$	-89.06%
$P_{MCI-AD-2}$	-99.41%

Table 6.16: Summary of Section 6.4.4

Then, we adopted *c*MPED and we motivated our choice. To show both the orthogonality of our approach to the choice of the measure for edge weights and to further investigate about the adequacy of *c*MPED, we repeated all the tasks described in Section 6.4.4 by substituting *c*MPED with PDI. After performing all tasks we obtained that the decrease of the connection coefficient when passing from $\mathcal{N}_{0_\pi}$ to $\mathcal{N}_{1_\pi}$ is -60.66 and -53.44 for the two MCI-MCI patients (it was -25.00 and -4.96 with *c*MPED), and -95.65 and -99.17 for the two MCI-AD patients (it was -89.06 and -99.41 with *c*MPED).

These results allow us to draw two important considerations, namely that: *(i)* *c*MPED is well suited since it behaved even better than PDI in the cases we have considered, and *(ii)* the general behavior of our approach is really orthogonal to the choice of the measure used for edge weights; in fact, it obtains satisfying results (even if worse) even when PDI is used instead of *c*MPED.

As a last remark about our approach, we observe that, in all the reasonings throughout this paper, we have considered the overall EEG tracing. However, it is well known that each EEG can be decomposed in several sub-bands (e.g., α, β, δ and θ) and that the analysis of these sub-bands can provide extremely useful information in the investigation of several neurological disorders. As for our approach, it can be applied either on the overall EEG or onto each of its sub-bands. Thanks to the analysis of sub-bands, further information useful for neurologists could be extracted.

Chapter 7

Conclusion

In this thesis we formally defined a framework that generalizes most of the existing string metrics based on symbol identity, making them suitable for application scenarios in which involved strings could be based on heterogeneous alphabets. We have shown how the natural assumption of symbol identity is strictly reductive and that *many-to-many* mappings between them need to be considered and handled. Due to these necessities, we based the framework on two main components that are a *matching schema*, which formalizes matches between symbols, and a *generalized metric function*, which abstracts the computation of string metrics on the basis of a pre-defined matching schema. We introduced the Multi-Parameterized Edit Distance, a generalization of the edit distance with the support of the proposed framework, and we discussed its computational issues. Due to the hardness of the MPED, various heuristics have been analyzed and developed: we started by making an excursus of various families of heuristics, then we focused in detail on few of them, in particular *hill climbing*, *simulated annealing* and *evolution strategy*. Finally, various application scenarios have been presented in which MPED has been successfully applied, possibly in conjunction with other various techniques, such as network analysis.

This work paved the way to a future in which numerous improvements of MPED and completely new application contexts can be researched. Some interesting examples are the following:

- *Multivariate time series analysis*: a multivariate time series [56] is a series of observations $x_i(t)$, $i = 1, \dots, n$, $t = 1, \dots, m$ made through time in sequence where i indicates the measurements made at each time point t and at least two measurements are present. Multivariate time series are common in various fields, such as finance and medicine, and by contrast to simple time series, they have not been extensively researched. Various challenges are still open, such as the definition of a general similarity measure for multivariate time series. For example, think of an object, such as an IoT device, carrying few sensors, each of which generates a number of values each instant of time. An example of query in this context would be that of find frequent usage patterns, in order to minimize the power consumption. An approach using MPED could exploit a common matching schema between all of the observations, in order to discover hidden correlations.

- *A formal language for constraints* χ: when the cardinalities of the alphabets Π_1 and Π_2 become large or the application context becomes complex (such as the multivariate time series), the definition of χ might become difficult. Thus, a formal language, with enough expressive power, could be useful in order to better define χ through a set of rules and a reasoning schema could be defined to navigate through valid matching schemas.

- *High performance computing for MPED*: to improve heuristics performance, the parallelization of its computation could be exploit. As a matter of fact, heuristics such as evolution strategy are highly prone to be parallelized [72] and at the time of writing this thesis, an evaluation of the effectiveness of parallelization is an ongoing work.

Bibliography

[1] Affenzeller, M. and Wagner, S. (2005). *Offspring Selection: A New Self-Adaptive Selection Scheme for Genetic Algorithms*, pages 218–221. Springer Vienna, Vienna. 56

[2] Ahmed, T., Coates, M., and Lakhina, A. (2007). Multivariate online anomaly detection using kernel recursive least squares. In *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pages 625–633. 123

[3] Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks*, 38(4):393 – 422. 122

[4] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410. 31

[5] Amir, A. and Nor, I. (2007). Generalized function matching. *Journal of Discrete Algorithms*, 5(3):514 – 523. Selected papers from Ad Hoc Now 2005. 30

[6] Apostolico, A., Erdős, P. L., and Lewenstein, M. (2007). Parameterized matching with mismatches. *Journal of Discrete Algorithms*, 5(1):135 – 140. 29

[7] Arvo, J. (2013). *Graphics gems II*. Elsevier. 133

[8] Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, Oxford, UK. 82

[9] Bäck, T., Foussette, C., and Krause, P. (2013). *Contemporary evolution strategies*. Springer. 86

[10] Baker, B. S. (1996). Parameterized pattern matching: Algorithms and applications. *Journal of Computer and System Sciences*, 52(1):28 – 42. 29, 31, 36, 40

[11] Baker, B. S. (1999). Parameterized diff. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '99, pages 854–855, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics. 30

[12] Basser, P. J., Mattiello, J., and LeBihan, D. (1994). Mr diffusion tensor spectroscopy and imaging. *Biophysical journal*, 66(1):259–267. 130

[13] Behrens, T., Berg, H. J., Jbabdi, S., Rushworth, M., and Woolrich, M. (2007). Probabilistic diffusion tractography with multiple fibre orientations: What can we gain? *NeuroImage*, 34(1):144 – 155. 130

[14] Behrens, T., Woolrich, M., Jenkinson, M., Johansen-Berg, H., Nunes, R., Clare, S., Matthews, P., Brady, J., and Smith, S. (2003). Characterization and propagation of uncertainty in diffusion-weighted mr imaging. *Magnetic Resonance in Medicine*, 50(5):1077–1088. 130

[15] Beyer, H.-G. and Schwefel, H.-P. (2002). Evolution strategies – a comprehensive introduction. *Natural Computing*, 1(1):3–52. 85

[16] Blin, G., Jiang, M., and Vialette, S. (2012). *The Longest Common Subsequence Problem with Crossing-Free Arc-Annotated Sequences*, pages 130–142. Springer Berlin Heidelberg, Berlin, Heidelberg. 30

[17] Bosman, H. H. W. J., Liotta, A., Iacca, G., and Wörtche, H. J. (2013a). Anomaly detection in sensor systems using lightweight machine learning. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 7–13. 122

[18] Bosman, H. H. W. J., Liotta, A., Iacca, G., and Wörtche, H. J. (2013b). Online extreme learning on fixed-point sensor networks. In *2013 IEEE 13th International Conference on Data Mining Workshops*, pages 319–326. 122, 123

[19] Cai, X., Nie, F., and Huang, H. (2013). Multi-view k-means clustering on big data. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 2598–2604. AAAI Press. 131

[20] Cao, Y., Tung, W., Gao, J., Protopopescu, V., and Hively, L. (2004). Detecting dynamical changes in time series using the permutation entropy. *Physical Review E*, 70:046217. 158

[21] Caruyer, E., Daducci, A., Descoteaux, M., Houde, J.-C., Thiran, J.-P., and Verma, R. (2014). Phantomas: a flexible software library to simulate diffusion mr phantoms. In *ISMRM*. 140, 144, 153, 155

[22] Catani, M. and De Schotten, M. T. (2008). A diffusion tensor imaging tractography atlas for virtual in vivo dissections. *Cortex*, 44(8):1105–1132. 130

[23] Cauteruccio, F., Fortino, G., Guerrieri, A., and Terracina, G. (2014). Discovery of hidden correlations between heterogeneous wireless sensor data streams. In *Internet and Distributed Computing Systems - 7th International Conference, IDCS 2014, Calabria, Italy, September 22-24, 2014. Proceedings*, pages 383–395. 52, 121

[24] Cauteruccio, F., Lo Giudice, P., Terracina, G., and Ursino, D. (2017a). Applying network analysis for extracting knowledge about environment changes from heterogeneous sensor data streams. In *26th Italian Workshop on Neural Networks*. 121

[25] Cauteruccio, F., Lo Giudice, P., Terracina, G., Ursino, D., Mammone, N., and Morabito, F. C. (2017b). A new string-based and network-based approach to investigating neurological disorders. *Submitted for publication. Available from the Authors.* 52, 121, 122

[26] Cauteruccio, F., Stamile, C., Terracina, G., Ursino, D., and Sappey-Marinier, D. (2015). An automated string-based approach to white matter fiber-bundles clustering. In *2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-17, 2015*, pages 1–8. 52, 121, 122

[27] Cauteruccio, F., Stamile, C., Terracina, G., Ursino, D., and Sappey-Marinier, D. (2016a). An automated string-based approach to extracting and characterizing white matter fiber-bundles. *Computers in Biology and Medicine*, 77:64–75. 52, 121, 122, 159

[28] Cauteruccio, F., Stamile, C., Terracina, G., Ursino, D., and Sappey-Marinier, D. (2016b). Improving quickbundles to extract anatomically coherent white matter fiber-bundles. In *Image Analysis and Recognition - 13th International Conference, ICIAR 2016, in Memory of Mohamed Kamel, Póvoa de Varzim, Portugal, July 13-15, 2016, Proceedings*, pages 633–641. 121, 122

[29] Cauteruccio, F., Stamile, C., Terracina, G., Ursino, D., and Sappey-Marinier, D. (2018). *Integrating QuickBundles into a Model-Guided Approach for Extracting "Anatomically-Coherent" and "Symmetry-Aware" White Matter Fiber-Bundles*, pages 39–46. Springer International Publishing, Cham. 121, 122

[30] Cauteruccio, F., Terracina, G., and Ursino, D. (2016c). Generalizing identity-based string comparison metrics: Framework and techniques. *Submitted for publication. Available from the Authors.* 33, 41, 52, 121

[31] Cauteruccio, F., Terracina, G., and Ursino, D. (2017c). Algorithms for strings and sequences: Searching motifs. In *Encyclopedia of Bioinformatics and Computational Biology*. Elsevier. To appear. 121

[32] Chan, K. and Fu, A. (1999). Efficient time series matching by wavelets. In *Proc. of the 15th IEEE International Conference on Data Engineering (ICDE'99)*, pages 126–133, Sydney, Australia. IEEE Computer Society Press. 163

[33] Cohen, W. W. (1998). Integration of heterogeneous databases without common domains using queries based on textual similarity. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD '98, pages 201–212, New York, NY, USA. ACM. 39

[34] Davatzikos, C., Bhatt, P., Shaw, L., Batmanghelich, K., and Trojanowski, J. (2011). Prediction of MCI to AD conversion, via MRI, CSF biomarkers, and pattern classification. *Neurobiology of aging*, 32(12):2322–e19. Elsevier. 175

[35] de Farias, C. M., Pirmez, L., Delicato, F. C., Pires, P. F., Guerrieri, A., Fortino, G., Cauteruccio, F., and Terracina, G. (2017). A multisensor data fusion algorithm using the hidden correlations in multiapplication wireless sensor data streams. In *14th IEEE International Conference on Networking, Sensing and Control, ICNSC 2017, Calabria, Italy, May 16-18, 2017*, pages 96–102. 121

[36] de Haan, W., Pijnenburg, Y., Strijers, R., van der Made, Y., van der Flier, W., Scheltens, P., and Stam, C. (2009). Functional neural network analysis in frontotemporal dementia and Alzheimer's disease using EEG and graph theory. *BMC neuroscience*, 10(1):1. BioMed Central. 158

[37] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38. 137

[38] Douek, P., Turner, R., Pekar, J., Patronas, N., and Le Bihan, D. (1991). Mr color mapping of myelin fiber orientation. *J Comput Assist Tomogr*, 15(6):923–929. 132

[39] Durbin, R., Eddy, S. R., Krogh, A., and Mitchison, G. (1998). *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press. 31

[40] Duun-Henriksen, J., Kjaer, T., Madsen, R., Remvig, L., Thomsen, C., and Sorensen, H. (2012a). Channel selection for automatic seizure detection. *Clinical Neurophysiology*, 123(1):84–92. 170

[41] Duun-Henriksen, J., Madsen, R., Remvig, L., Thomsen, C., Sorensen, H., and Kjaer, T. (2012b). Automatic detection of childhood absence epilepsy seizures: toward a monitoring device. *Pediatric Neurology*, 46(5):287–292. 157, 170

[42] Eiben, A. E., Smith, J. E., et al. (2003). *Introduction to evolutionary computing*, volume 53. Springer. 82

[43] Elmagarmid, A. K., Ipeirotis, P. G., and Verykios, V. S. (2007). Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16. 28

[44] Fernau, H. and Schmid, M. L. (2015). Pattern matching with variables: A multivariate complexity analysis. *Information and Computation*, 242(Supplement C):287 – 305. 30

[45] Fortino, G., Giannantonio, R., Gravina, R., Kuryloski, P., and Jafari, R. (2013). Enabling effective programming and flexible management of efficient body sensor network applications. *IEEE Transactions on Human-Machine Systems*, 43(1):115–133. 122

[46] Fortino, G., Guerrieri, A., O'Hare, G., and Ruzzelli, A. (2012). A flexible building management framework based on wireless sensor and actuator networks. *Journal of Network and Computer Applications*, 35(6):1934 – 1952. 122, 124

[47] García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., and Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1):18 – 28. 123

[48] Garey, M. R. and Johnson, D. S. (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA. 45

[49] Garyfallidis, E., Brett, M., Correia, M. M., Williams, G. B., and Nimmo-Smith, I. (2012). Quickbundles, a method for tractography simplification. *Frontiers in neuroscience*, 6. 130, 150, 151

[50] Gawrychowski, P. and Uznański, P. (2016). Order-preserving pattern matching with k mismatches. *Theoretical Computer Science*, 638(Supplement C):136 – 144. Pattern Matching, Text Data Structures and Compression. 31

[51] Gnawali, O., Fonseca, R., Jamieson, K., Kazandjieva, M., Moss, D., and Levis, P. (2013). Ctp: An efficient, robust, and reliable collection tree protocol for wireless sensor networks. *ACM Trans. Sen. Netw.*, 10(1):16:1–16:49. 124

[52] Golby, A. J., Kindlmann, G., Norton, I., Yarmarkovich, A., Pieper, S., and Kikinis, R. (2011). Interactive diffusion tensor tractography visualization for neurosurgical planning. *Neurosurgery*, 68(2):496–505. 130

[53] Gravano, L., Ipeirotis, P. G., Koudas, N., and Srivastava, D. (2003). Text joins in an rdbms for web data integration. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, pages 90–101, New York, NY, USA. ACM. 39

[54] Greco, G. and Terracina, G. (2013). Frequency-based similarity for parameterized sequences: Formal framework, algorithms, and applications. *Information Sciences*, 237(Supplement C):176 – 195. Prediction, Control and Diagnosis using Advanced Neural Computations. 31

[55] Hamming, R. W. (1950). Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160. 26, 27

[56] Harvey, A. C. and Harvey, A. C. (1993). *Time series models*, volume 2. Harvester Wheatsheaf New York. 181

[57] Hazay, C., Lewenstein, M., and Sokol, D. (2007). Approximate parameterized matching. *ACM Trans. Algorithms*, 3(3). 29

[58] Hebert, L., Weuve, J., Scherr, P., and Evans, D. (2013). Alzheimer disease in the United States (2010–2050) estimated using the 2010 census. *Neurology*, 80(19):1778–1783. AAN Enterprises. 171

[59] Heckbert, P. (1982). Color image quantization for frame buffer display. *SIGGRAPH Comput. Graph.*, 16(3):297–307. 133

[60] Henikoff, S. and Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919. 31

[61] Hornby, G. S. (2006). Alps: The age-layered population structure for reducing the problem of premature convergence. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, pages 815–822, New York, NY, USA. ACM. 56

[62] Hornero, R., Abásolo, D., Escudero, J., and Gómez, C. (2009). Nonlinear analysis of electroencephalogram and magnetoencephalogram recordings in patients with Alzheimer's disease. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 367(1887):317–336. The Royal Society. 157, 171

[63] Hua, K., Zhang, J., Wakana, S., Jiang, H., Li, X., Reich, D. S., Calabresi, P. A., Pekar, J. J., van Zijl, P. C., and Mori, S. (2008). Tract probability maps in stereotaxic spaces: analyses of white matter anatomy and tract-specific quantification. *Neuroimage*, 39(1):336–347. 130

[64] Hunter, J. and Colley, M. (2007). Feature Extraction from Sensor Data Streams for Real-Time Human Behaviour Recognition. In *PKDD*, volume 4702 of *Lecture Notes in Computer Science*, pages 115–126. Springer. 159

[65] Jaro, M. A. (1978). *Unimatch: A record linkage system: Users manual*. Bureau of the Census. 28, 38

[66] Jeong, J. (2004). EEG dynamics in patients with Alzheimer's disease. *Clinical neurophysiology*, 115(7):1490–1505. Elsevier. 158, 171

[67] Kamiya, T., Kusumoto, S., and Inoue, K. (2002). Ccfinder: a multilinguistic token-based code clone detection system for large scale source code. *IEEE Transactions on Software Engineering*, 28(7):654–670. 20, 29, 30

[68] Kaufman, L. and Rousseeuw, P. J. (1987). Clustering by means of medoids. 137

[69] Keller, O., Kopelowitz, T., and Lewenstein, M. (2009). On the longest common parameterized subsequence. *Theoretical Computer Science*, 410(51):5347 – 5353. Combinatorial Pattern Matching. 30

[70] Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrotra, S. (2001). Dimensionality reduction for fast similarity search in large time series databases. *Journal of Knowledge and Information Systems*, 3(3):263–286. 163

[71] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680. 55, 75

[72] Kumar, V. (2002). *Introduction to Parallel Computing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition. 182

[73] Landau, G. M. and Vishkin, U. (1989). Fast parallel and serial approximate string matching. *Journal of Algorithms*, 10(2):157 – 169. 71

[74] Le Bihan, D., Mangin, J.-F., Poupon, C., Clark, C. A., Pappata, S., Molko, N., and Chabriat, H. (2001). Diffusion tensor imaging: concepts and applications. *Journal of magnetic resonance imaging*, 13(4):534–546. 132, 144

[75] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. 27, 37, 159

[76] Levis, P., Patel, N., Culler, D., and Shenker, S. (2004). Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *Proceedings of the 1st Conference on Symposium on Networked Systems Design and Implementation - Volume 1*, NSDI'04, pages 2–2, Berkeley, CA, USA. USENIX Association. 124

[77] Lipsky, O., Porat, B., Porat, E., Shalom, B. R., and Tzur, A. (2010). String matching with up to k swaps and mismatches. *Information and Computation*, 208(9):1020 – 1030. 29

[78] Lo Giudice, P., Mammone, N., Morabito, F., Ursino, D., Aguglia, U., Cianci, V., Ferlazzo, E., and Gasparini, S. (2017). Usage of network analysis to investigate Periodic Sharp Wave Complexes in EEGs of patients with sporadic CJD. In *Atti del Venticinquesimo Convegno Nazionale su Sistemi Evoluti per Basi di Dati (SEBD'17)*, Squillace (CZ), Italy. 158

[79] MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA. 137

[80] Mammone, N., Bonanno, L., Salvo, S. D., Marino, S., Bramanti, P., Bramanti, A., and Morabito, F. (In press). Permutation Disalignment Index as an Indirect, EEG-Based, Measure of Brain Connectivity in MCI and AD Patients. *International Journal of Neural Systems*, http://dx.doi.org/10.1142/S0129065717500204. xiii, 177, 178, 179

[81] Mammone, N., Foresta, F. L., and Morabito, F. (2011). Discovering network phenomena in the epileptic electroencephalography through permutation entropy mapping. *Frontiers in Artificial Intelligence and Applications*, Neural Nets WIRN10:260–269. IOS Press. 158

[82] Mårtensson, J., Nilsson, M., Ståhlberg, F., Sundgren, P. C., Nilsson, C., van Westen, D., Larsson, E.-M., and Lätt, J. (2013). Spatial analysis of diffusion tensor tractography statistics along the inferior fronto-occipital fasciculus with application in progressive supranuclear palsy. *Magnetic Resonance Materials in Physics, Biology and Medicine*, 26(6):527–537. 130

[83] Mendivelso, J. and Pinzón, Y. (2015). Parameterized matching: Solutions and extensions. In Holub, J. and Žďárek, J., editors, *Proceedings of the Prague Stringology Conference 2015*, pages 118–131, Czech Technical University in Prague, Czech Republic. 30

[84] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092. 75

[85] Mitchell, M. (1998). *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA. 55

[86] Monge, A. E. and Elkan, C. P. (1996). The field matching problem: Algorithms and applications. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 267–270. AAAI Press. 38

[87] Mori, S., Crain, B. J., Chacko, V., and Van Zijl, P. (1999). Three-dimensional tracking of axonal projections in the brain by magnetic resonance imaging. *Annals of neurology*, 45(2):265–269. 130

[88] Ordóñez, P. et al. (2008). Visualizing multivariate time series data to detect specific medical conditions. In *AMIA Annual Symposium Proceedings*, volume 2008, page 530. American Medical Informatics Association. 148

[89] Ouyang, G., Li, J., Liu, X., and Li, X. (2013). Dynamic characteristics of absence EEG recordings with multiscale permutation entropy analysis. *Epilepsy Research*, 104(3):246–252. 158

[90] O'Donnell, L., Kubicki, M., Shenton, M. E., Dreusicke, M. H., Grimson, W. E. L., and Westin, C.-F. (2006). A method for clustering white matter fiber tracts. *American Journal of Neuroradiology*, 27(5):1032–1036. 130

[91] Pantano, P., Mainero, C., Iannetti, G. D., Caramia, F., Legge, S. D., Piattella, M. C., Pozzilli, C., Bozzao, L., and Lenzi, G. L. (2002). Contribution of corticospinal tract damage to cortical motor reorganization after a single clinical attack of multiple sclerosis. *NeuroImage*, 17(4):1837 – 1843. 130

[92] Peng, Y.-H. and Yang, C.-B. (2014). Finding the gapped longest common subsequence by incremental suffix maximum queries. *Information and Computation*, 237(Supplement C):95 – 100. 30

[93] Petersen, R. (2004). Mild cognitive impairment as a diagnostic entity. *Journal of internal medicine*, 256(3):183–194. Wiley Online Library. 175

[94] Phua, C., Lee, V., Smith, K., and Gayler, R. (2010). A comprehensive survey of data mining-based fraud detection research. *arXiv preprint arXiv:1009.6119*. 123

[95] Ponten, S., Douw, L., Bartolomei, F., Reijneveld, J., and Stam, C. (2009). Indications for network regularization during absence seizures: weighted and unweighted graph theoretical analyses. *Experimental Neurology*, 217(1):197–204. Elsevier. 158, 170

[96] Powers, D. M. (2011). Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 142, 145

[97] Ramirez, J., Gorriz, J., Salas-Gonzalez, D., Romero, A., Lopez, M., Alvarez, I., and Gomez-Rio, M. (2013). Computer-aided diagnosis of Alzheimer?s type dementia combining support vector machines and discriminant set of features. *Information Sciences*, 237:59–72. Elsevier. 171

[98] Rechenberg, I. (1965). Cybernetic solution path of an experimental problem. 86

[99] Sant'Anna, A., Salarian, A., and Wickström, N. (2011). A New Measure of Movement Symmetry in Early Parkinson's Disease Patients Using Symbolic Processing of Inertial Sensor Data. *IEEE Trans. Biomed. Engineering*, 58(7):2127–2135. 159

[100] Schwarz, R., Dayhoff, M., and Orcutt, B. (1978). A model of evolutionary change in proteins. *Atlas of protein sequence structures*, pages 345–352. 31

[101] Schwefel, H.-P. (1968). Experimentelle optimierung einer zweiphasendüse. *Interner Bericht HE/F*. 86

[102] Shie, B., Jang, F., and Tseng, V. (2010). Intelligent panic disorder treatment by using biofeedback analysis and web technologies. *IJBIDM*, 5(1):77–93. 159

[103] Simon, D. (2013). *Evolutionary optimization algorithms*. John Wiley & Sons. 82, 87

[104] Smith, T. and Waterman, M. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195 – 197. 38

[105] Stam, C., Haan, W. D., Daffertshofer, A., Jones, B., Manshanden, I., Walsum, A. V., Montez, T., Verbunt, J., Munck, J. D., Dijk, B. V., et al. (2009). Graph theoretical analysis of magnetoencephalographic functional connectivity in Alzheimer's disease. *Brain*, 132(1):213–224. Oxford Univ Press. 158

[106] Stam, C., Made, Y. V. D., Pijnenburg, Y., and Scheltens, P. (2003). EEG synchronization in mild cognitive impairment and Alzheimer's disease. *Acta Neurologica Scandinavica*, 108(2):90–96. Wiley Online Library. 158

[107] Stamile, C., Cauteruccio, F., Terracina, G., Ursino, D., Kocevar, G., and Sappey-Marinier, D. (2015). A model-guided string-based approach to white matter fiber-bundles extraction. In *Brain Informatics and Health - 8th International Conference, BIH 2015, London, UK, August 30 - September 2, 2015. Proceedings*, pages 135–144. 52, 121, 122

[108] Stankovic, J. A. (2008). When sensor and actuator networks cover the world. *ETRI Journal*, 30(5):627–633. 122

[109] Steinhoff, B., Zerr, I., Glatting, M., Schulz-Schaeffer, W., Poser, S., and Kretzschmar, H. (2004). Diagnostic value of periodic complexes in Creutzfeldt–Jakob disease. *Annals of Neurology*, 56(5):702–708. Wiley Online Library. 158, 167

[110] Sun, S. (2013). A survey of multi-view machine learning. *Neural Computing and Applications*, 23(7):2031–2038. 131

[111] Tournier, J.-D., Calamante, F., and Connelly, A. (2012). Mrtrix: Diffusion tractography in crossing fiber regions. *Int. J. Imaging Syst. Technol.*, 22(1):53–66. 148

[112] Traub, R. and Pedley, T. (1981). Virus-induced electrotonic coupling: Hypothesis on the mechanism of periodic EEG discharges in Creutzfeldt-Jakob disease. *Annals of Neurology*, 10(5):405–410. Wiley Online Library. 167

[113] Wagner, R. A. and Fischer, M. J. (1974). The string-to-string correction problem. *J. ACM*, 21(1):168–173. 28

[114] Wagner, S., Kronberger, G., Beham, A., Kommenda, M., Scheibenpflug, A., Pitzer, E., Vonolfen, S., Kofler, M., Winkler, S., Dorfer, V., and Affenzeller, M. (2014). *Advanced Methods and Applications in Computational Intelligence*, volume 6 of *Topics in Intelligent Engineering and Informatics*, chapter Architecture and Design of the HeuristicLab Optimization Environment, pages 197–261. Springer. 53, 55

[115] Wang, P., Wu, Y., Hung, C., Kwan, S., Teng, S., and Soong, B. (2008). Early detection of periodic sharp wave complexes on EEG by independent component analysis in patients

with Creutzfeldt-Jakob disease. *Journal of Clinical Neurophysiology*, 25(1):25–31. LWW. 158, 167

[116] Waterman, M., Smith, T., and Beyer, W. (1976). Some biological sequence metrics. *Advances in Mathematics*, 20(3):367 – 387. 37

[117] (WHO), W. H. O. (1998). Consensus on criteria for diagnosis of sporadic CJD. *Weekly Epidemiological Record*, 73:361–365. 167

[118] Wieser, H., Schindler, K., and Zumsteg, D. (2006). EEG in Creutzfeldt–Jakob disease. *Clinical Neurophysiology*, 117(5):935–951. Elsevier. 157, 167

[119] Wieser, H., Schwarz, U., Blättler, T., Bernoulli, C., Sitzler, M., Stoeck, K., and Glatzel, M. (2004). Serial EEG findings in sporadic and iatrogenic Creutzfeldt–Jakob disease. *Clinical Neurophysiology*, 115(11):2467–2478. Elsevier. 167

[120] Wilson, M., Tench, C., Morgan, P., and Blumhardt, L. (2003). Pyramidal tract mapping by diffusion tensor magnetic resonance imaging in multiple sclerosis: improving correlations with disability. *J Neurol Neurosurg Psychiatry*, 74(2):203–207. 130

[121] Yeatman, J. D., Dougherty, R. F., Myall, N. J., Wandell, B. A., and Feldman, H. M. (2012). Tract profiles of white matter properties: automating fiber-tract quantification. *PloS one*, 7(11):e49790. 130

[122] Yendiki, A., Panneck, P., Srinivasan, P., Stevens, A., Zöllei, L., Augustinack, J., Wang, R., Salat, D. H., Ehrlich, S., Behrens, T. E. J., Jbabdi, S., Gollub, R. L., and Fischl, B. (2011). Automated probabilistic reconstruction of white-matter pathways in health and disease using an atlas of the underlying anatomy. *Front. Neuroinform.*, 2011. 130

[123] Yi, B. and Faloutos, C. (2000). Fast time sequence indexing for arbitrary lp norms. In *Proceedings of the 26th International Conference on Very Large Databases*, pages 385–394, Cairo, Egypt. 163

[124] Yu, M., Gouw, A., Hillebrand, A., Tijms, B., Stam, C., van Straaten, E., and Pijnenburg, Y. (2016). Different functional connectivity and network topology in behavioral variant of frontotemporal dementia and Alzheimer's disease: an EEG study. *Neurobiology of aging*, 42:150–162. Elsevier. 158

[125] Zhang, S., Correia, S., and Laidlaw, D. H. (2008). Identifying white-matter fiber bundles in dti data using an automated proximity-based fiber-clustering method. *IEEE transactions on visualization and computer graphics*, 14(5):1044–1053. 130

[126] Zhang, Y. and Jiang, J. (2008). Bibliographical review on reconfigurable fault-tolerant control systems. *Annual Reviews in Control*, 32(2):229 – 252. 123

[127] Zoumboulakis, M. and Roussos, G. (2011). Complex Event Detection in Extremely Resource-Constrained Wireless Sensor Networks. *MONET*, 16(2):194–213. 159