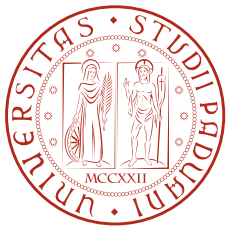


Università degli Studi di Padova

Scuola di Dottorato di Ricerca in
Ingegneria dell'Informazione

Tesi di Dottorato

Advanced Computational Methods for Massive Biological Sequence Analysis

Direttore della Scuola

Ch.mo Prof. Matteo Bertocco

Supervisore

Dott. Matteo Comin

Dottorando

Davide Verzotto

Ciclo XXIV, Anno 2011

Abstract

With the advent of modern sequencing technologies massive amounts of biological data, from protein sequences to entire genomes, are becoming increasingly available. This poses the need for the automatic analysis and classification of such a huge collection of data, in order to enhance knowledge in the Life Sciences. Although many research efforts have been made to mathematically model this information, for example finding patterns and similarities among protein or genome sequences, these approaches often lack structures that address specific biological issues.

In this thesis, we present novel computational methods for three fundamental problems in molecular biology: the detection of remote evolutionary relationships among protein sequences; the identification of subtle biological signals in related genome or protein functional sites; and the phylogeny reconstruction by means of whole-genome comparisons. The main contribution is given by a systematic analysis of patterns that may affect these tasks, leading to the design of practical and efficient new pattern discovery tools. We thus introduce two advanced paradigms of pattern discovery and filtering based on the insight that functional and conserved biological motifs, or patterns, should lie in different sites of sequences. This enables to carry out space-conscious approaches that avoid a multiple counting of the same patterns.

The first paradigm considered, namely *irredundant common motifs*, concerns the discovery of common patterns, for two sequences, that have occurrences not covered by other patterns, whose coverage is defined by means of specificity and extension. The second paradigm, namely *underlying motifs*, concerns the filtering of patterns, from a given set, that have occurrences not overlapping other patterns with higher priority, where priority is defined

by lexicographic properties of patterns on the boundary between pattern matching and statistical analysis. We develop three practical methods directly based on these advanced paradigms. Experimental results indicate that we are able to identify subtle similarities among biological sequences, using the same type of information only once.

In particular, we employ the irredundant common motifs and the statistics based on these patterns to solve the remote protein homology detection problem. Results show that our approach, called Irredundant Class, outperforms the state-of-the-art methods in a challenging benchmark for protein analysis. Afterwards, we establish how to compare and filter a large number of complex motifs (e.g., degenerate motifs) obtained from modern motif discovery tools, in order to identify subtle signals in different biological contexts. In this case we employ the notion of underlying motifs. Tests on large protein families indicate that we drastically reduce the number of motifs that scientists should manually inspect, further highlighting the actual functional motifs. Finally, we combine the two proposed paradigms to allow the comparison of whole genomes, and thus the construction of a novel and practical distance function. With our method, called Unic Subword Approach, we relate to each other the regions of two genome sequences by selecting conserved motifs during evolution. Experimental results show that our approach achieves better performance than other state-of-the-art methods in the whole-genome phylogeny reconstruction of viruses, prokaryotes, and unicellular eukaryotes, further identifying the major clades of these organisms.

Sommario

Con l'avvento delle moderne tecnologie di sequenziamento, massive quantità di dati biologici, da sequenze proteiche fino a interi genomi, sono disponibili per la ricerca. Questo progresso richiede l'analisi e la classificazione automatica di tali collezioni di dati, al fine di migliorare la conoscenza nel campo delle Scienze della Vita. Nonostante finora siano stati proposti molti approcci per modellare matematicamente le sequenze biologiche, ad esempio cercando pattern e similarità tra genomi o proteine, questi metodi spesso mancano di strutture in grado di indirizzare specifiche questioni biologiche.

In questa tesi, presentiamo nuovi metodi computazionali per tre problemi fondamentali della biologia molecolare: la scoperta di relazioni evolutive remote tra sequenze proteiche; l'individuazione di segnali biologici complessi in siti funzionali tra loro correlati; e la ricostruzione della filogenesi di un insieme di organismi, attraverso la comparazione di interi genomi. Il principale contributo è dato dall'analisi sistematica dei pattern che possono interessare questi problemi, portando alla progettazione di nuovi strumenti computazionali efficaci ed efficienti. Vengono introdotti così due paradigmi avanzati per la scoperta e il filtraggio di pattern, basati sull'osservazione che i motivi biologici funzionali, o pattern, sono localizzati in differenti regioni delle sequenze in esame. Questa osservazione consente di realizzare approcci parsimoniosi in grado di evitare un conteggio multiplo degli stessi pattern.

Il primo paradigma considerato, ovvero *irredundant common motifs*, riguarda la scoperta di pattern comuni a coppie di sequenze che hanno occorrenze non coperte da altri pattern, la cui copertura è definita da una maggiore specificità e/o possibile estensione dei pattern. Il secondo paradigma, ovvero *underlying motifs*, riguarda il filtraggio di pattern che hanno occorrenze non sovrapposte a quelle di altri pattern con maggiore priorità, dove la prio-

rità è definita da proprietà lessicografiche dei pattern al confine tra pattern matching e analisi statistica. Sono stati sviluppati tre metodi computazionali basati su questi paradigmi avanzati. I risultati sperimentali indicano che i nostri metodi sono in grado di identificare le principali similitudini tra sequenze biologiche, utilizzando l'informazione presente in maniera non ridondante.

In particolare, impiegando gli irredundant common motifs e le statistiche basate su questi pattern risolviamo il problema della rilevazione di omologie remote tra sequenze proteiche. I risultati evidenziano che il nostro approccio, chiamato *Irredundant Class*, ottiene ottime prestazioni su un benchmark impegnativo e migliora i metodi allo stato dell'arte. Inoltre, per individuare segnali biologici complessi utilizziamo la nozione di underlying motifs, definendo così alcune modalità per il confronto e il filtraggio di motivi degenerati ottenuti tramite moderni strumenti di pattern discovery. Esperimenti su grandi famiglie proteiche dimostrano che il nostro metodo riduce drasticamente il numero di motivi che gli scienziati dovrebbero altrimenti ispezionare manualmente, mettendo in luce inoltre i motivi funzionali identificati in letteratura. Infine, combinando i due paradigmi proposti presentiamo una nuova e pratica funzione di distanza tra interi genomi. Con il nostro metodo, chiamato *Unic Subword Approach*, relazioniamo tra loro le diverse regioni di due sequenze genomiche, selezionando i motivi conservati durante l'evoluzione. I risultati sperimentali evidenziano che il nostro approccio offre migliori prestazioni rispetto ad altri metodi allo stato dell'arte nella ricostruzione della filogenesi di organismi quali virus, procarioti ed eucarioti unicellulari, identificando inoltre le sottoclassi principali di queste specie.

To my awesome wife Chiara,
and my special family Mara, Lino e Marco

Acknowledgments

It has been more than eight years since I started my career at the University of Padova, a very long life experience. The last three of which were the most intense, with hard work to complete my doctorate degree and a research project ever growing. I feel particularly lucky to have had the opportunity to travel the world for part of this period; and, above all, to have had a beautiful wedding in Las Vegas with my beloved Chiara.

First, I would like to thank Matteo Comin for his advice, the many stimulating discussions, and for putting up with me throughout the last four years. As well as for the scientific collaborations that led to this Thesis.

A big thanks goes to Stefano Lonardi for having taken me under his wing during my great experience at the University of California, Riverside and his patience for the articles to be published.

My wife and my family have a major role in this Thesis, having supported and accepted what I have done so far, and even more by supporting what I do in the near future. My wife has always been with me during my Ph.D., and I am forever grateful for that. And even if I have been stressful at times, it was worth it.

I would like to thank all my close Ph.D. friends and dear fellows, in the lexicographic order: Anton, Cinzia, Ferdinando, Marco, Michele, Nicola, and Simone.

A particular thanks goes to Alessandra, and then to Madie, Cristina, Makbule, Roberta, and Sandra.

During the time spent doing research and teaching assistance, I have had the great opportunity to learn from and exchange ideas with: Alberto Apostolico, Franco Bombi, Concettina Guerra, Karine Le Roch, Adriano Luchetta, Cinzia Pizzi, and Luca Pretto; and then with: Gianfranco Cia-

rdo, Bruno Codenotti, Nello Cristianini, Carlo Ferrari, Tao Jiang, Giuseppe Lancia, Christina Leslie, and Nadia Ponts.

I would also mention all my other friends in the university sphere (I apologize to those who are not mentioned here): Alberto, Alessandro, Alex, Ana, Angela, Angie, Dalva, Damiano, David, Davide, Denise, Elisa, Emanuele (x 2), Elena H., Enrico, Francesco, Gianluca, Giovanni, Giovanni Maria, Giulia, Ismael, Jennifer, Jon, Jonathan, José, Li, Lucia, Nicola, Olga, Piotr, Sandra, Saverio, Tiziana, Wei, and Yuri.

Last but not least, I want to thank all my friends who have accompanied me every day: my brother Marco; Alessio, Emanuele, Luca, Michele, and Riccardo; and thus all my close relatives.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The increasing availability of massive biological sequences, from protein sequences to entire genomes, poses the need for the automatic analysis and classification of such a huge collection of data. Alignment methods and pattern discovery techniques have been used, for long time, to address various problems emerging in the field of computational molecular biology. Unfortunately many of these methods do not scale well with the length and complexity of the sequences under consideration, and therefore are not practical for applications in genome and protein analysis. To overcome this recent obstacle, a number of techniques which do not rely on alignment have been conceived; these methods are also called alignment-free methods. Although several alignment-free methods have been proposed over the years, the development of adequate computational tools able to classify and digest entire genomes and proteomes is still in its infancy.

1.1 A Biological Overview

Biomolecular sequences, i.e., protein primary structure and nucleic acids, namely DNA and RNA, represent the most basic type of biological information. Features of these sequences that are reused by Nature help us to better understand the basic mechanisms of gene structure, function, and regulation [115; 119]. Moreover, tools able to correctly classify this information allow us to study evolutionary mechanisms and to infer possible associations between

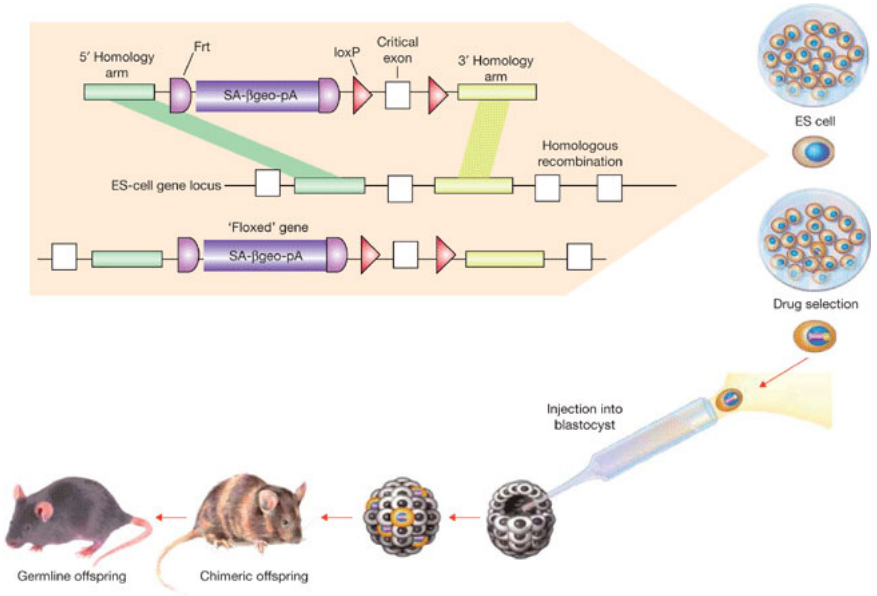


FIGURE 1.1. *Concept strategy for drug development through mouse genome mutations, as appeared in a 2007 article of Nature Cell Biology [97].*

different molecules. For example, we can generate more accurate models of human pathologies through the development of conditional and inducible mouse mutations [97]. In this regard biologists have recently discovered new drugs and therapies, such as antibiotics, vaccines, and cancer treatments, by looking at the correlations between particular human and mouse biomolecules at genome or protein levels, in order to find conserved functional sites that may be involved in a disease. Scientists can then act on individual chemicals, e.g., on mouse, and analyze the behavior of *in vivo* or *in vitro* cells in response to a specific drug. For instance, Figure 1.1 shows a strategy for the analysis of pathologies by means of genome recombinations. Indeed effects of drug are characterized by specific interactions of the created compound with other biomolecules of the target host, reinforcing the idea that some mechanism occur due to the association between different molecules inside the cell. We are particularly interested in these biological associations, or *patterns*.

DNA (deoxyribonucleic acid), RNA (ribonucleic acid), and proteins are essential for the development and functioning of all known forms of life. These

molecules are arranged in chains of chemical compounds, which can be seen as sequences based on different coding alphabets. For example, the DNA molecule consists of two long polymers of simple units, the nucleotides, each one linked to a backbone made by an alternating sugar-phosphate sequence. These chains have a direction, known as polarity 5' to 3', and are called DNA strands; they run in opposite directions to each other and join together through complementary nucleotide bases, or base pairs, in order to form the characteristic double helix. To have a more concrete idea about this concept, Figure 1.2 shows the DNA molecule packaged into a chromosome, inside the cell of a eukaryotic organism, where the the four bases characterizing the DNA sequence are well placed in evidence: adenine `A`, cytosine `C`, guanine `G`, and thymine `T`. Each base in one DNA strand, e.g., the forward, is paired to its complement in the opposite strand, the reverse, in this way: `[A, T]` form a class of complementary nucleotides, while `[C, G]` form the other class. The sequence of data given by these four bases encodes the necessary information for the functioning of an organism. The segments carrying the genetic information are called genes, and contain the biological instructions that make every living being unique. Part of genes are finally coded into RNA and proteins.

Like the DNA molecule, RNA is made up of a long polymer of nucleotides, consisting of the four bases `A,C,G,U`, whereas uracil `U` binds to adenine in place of thymine `T` found in DNA. In some case, like for viruses, the RNA could be the unique molecule available in the whole system that carries the genetic information, thus regulating its functioning. Proteins are coded with 20 amino acids, here denoted with the one-letter symbols: `A,R,N,D,C,Q,E,G,H,I,L,K,M,F,P,S,T,W,Y,V`. Throughout the thesis we will refer to the word *genome* as the complete set of genetic and non-genetic DNA/RNA material of a living organism, and with *proteome* the complete set of known proteins for a whole organism.

The central dogma of molecular biology permits us to better understands the transfer of information between different molecules, as indicated in Figure 1.3. This dogma states that the biological information can be transfered from DNA to RNA, and vice versa in a reverse transcription, and from RNA to proteins, but cannot be transferred back from proteins to ei-

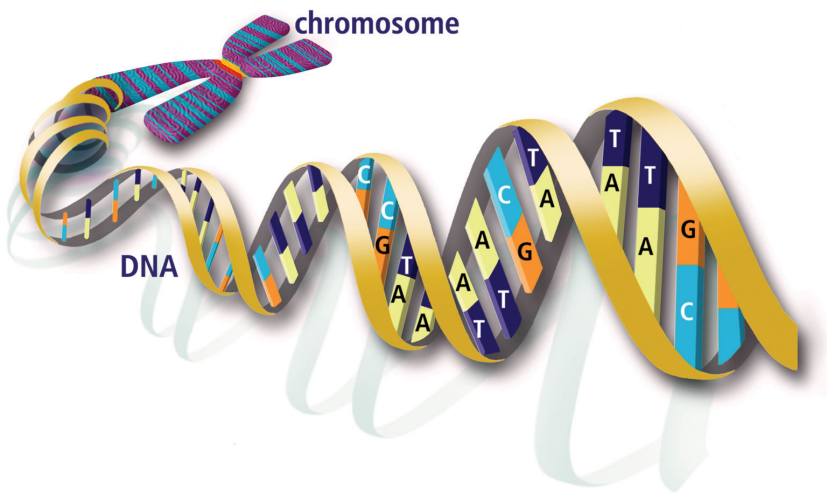


FIGURE 1.2. *DNA sequence packaged to form a chromosome, showing the complementary strands. The sequences obtained from all chromosomes of an organism give its whole genome (U.S. Biological and Environmental Research Information System, Oak Ridge National Laboratory).*

ther proteins or nucleic acids [31]. DNA and RNA genomes can replicate themselves, while proteins may interact with other proteins and with nucleic acids. Some of these latter interactions have been recently mapped into epigenetic frameworks, where particular groups of proteins packaging DNA, and called histones, may inhibit some parts of the genetic code during DNA-RNA transcription, or prevent modifications to DNA during replication [56]. This understanding has arisen through the development of next-generation sequencing technologies capable of producing genome-wide mapping of the genetic code [102].

In brief, transfers describe the normal flow of biological information in eukaryotic and prokaryotic species, as of Figure 1.4: DNA nucleotides can be copied into DNA (replication); well-defined segments of DNA, the coding regions, are copied into messenger RNA, or mRNA (transcription); there, proteins can be synthesized using part of the information contained in the mRNA, comprising an amino acid sequence specified by the genetic code (translation). Whenever a protein sequence is translated from nucleic acids, it

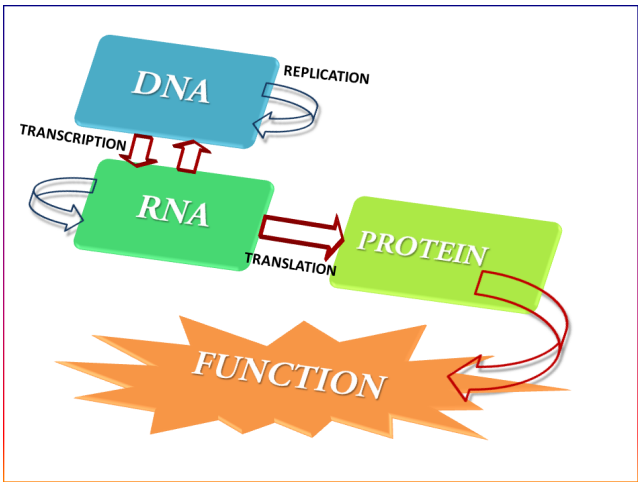


FIGURE 1.3. *The central dogma of molecular biology.*

adopts a specific three-dimensional conformation, or tertiary structure, that carries a particular chemical or physical function in the cell (see Figure 1.5). Protein coding regions are delimited within genes in DNA, and are characterized by specific biological functions once translated into proteins. Each gene is transcribed by means of enzymes that bind to particular regulatory regions of DNA, which are located upstream of the gene. We are interested in these regulatory elements, or transcription factor binding sites, that likely shape similar patterns and spatial arrangements in the DNA molecule for clusters of genes and different living organisms. The methods we will present in Chapter 3 can be explicitly used to identify such biological signals.

The analysis of biological datasets is in general a hard task inherently due to the combinatorial nature of the sequences and the level of information that scientists want to achieve. One of the best techniques so far known in computational molecular biology for the analysis of closely related molecules, is the sequence alignment [82; 109]. In short, the sequence alignment matches the symbols of two entire sequence data (global alignment) or part of them (local alignment), considering events that might have occurred during the evolution of the two sequences from a common ancestor. These events can be insertions, deletions, or modifications of nucleotide and amino acid residues. To each possible alignment, we assigned a score by means of a scheme that penalizes gaps (i.e., indels) and mismatches (i.e., modifications or substitu-

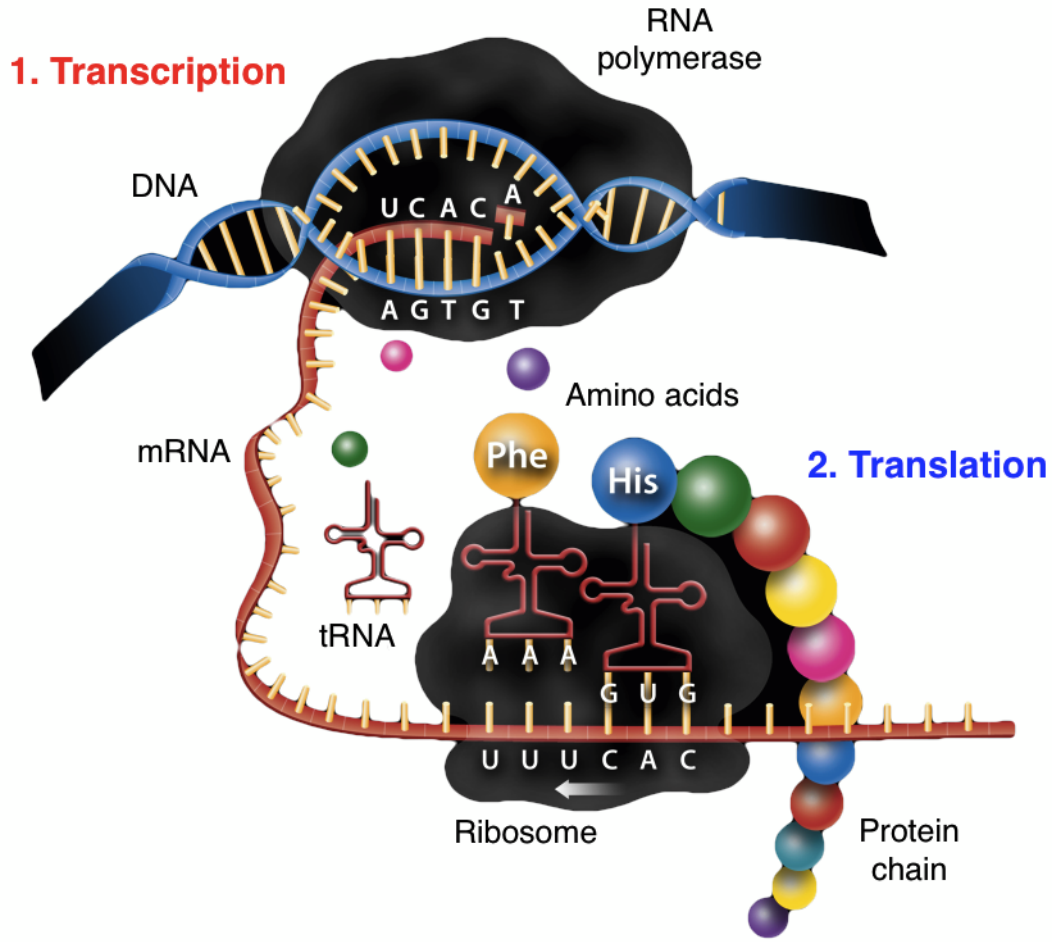


FIGURE 1.4. *Diagram of the main protein synthesis steps from genes of prokaryotic species (U.S. Biological and Environmental Research Information System, Oak Ridge National Laboratory).*

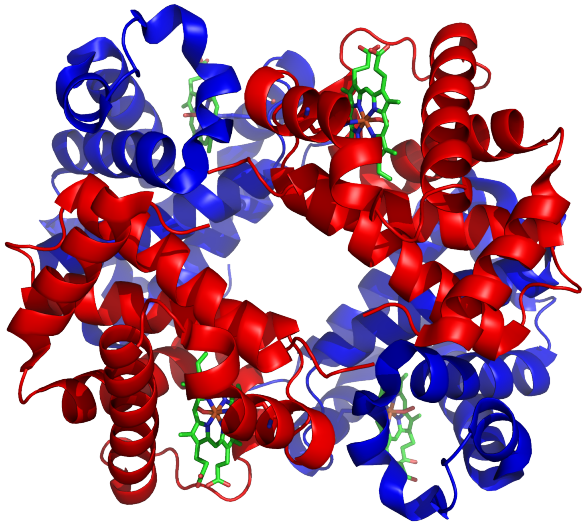


FIGURE 1.5. *Three-dimensional structure of human protein hemoglobin, the oxygen-carrying molecule in red blood cells of almost all vertebrates, taken from the RCSB Protein Data Bank.*

tions), usually with the help of certain substitution matrices, e.g., [50]. The resulting alignments having optimal score indeed identify regions of similarity among the sequences.

Because of the peculiarities of such schemes, alignment methods are unlikely to analyze large sets of sequences with low or sparse similarity. Moreover, the sequence alignment and the related multiple sequence alignment, a technique addressing the above issues, typically do not scale well with the size of most biological sequences, and therefore are not practical for common problems. Similarly, scientists working on massive data —from protein datasets, which typically comprise a large number of sequences, to entire genomes— require advanced computational tools to address specific questions in molecular biology. These approaches are also called alignment-free methods (see [116] for a comprehensive review).

1.2 Processing Massive Sequences

With the advent of fast and efficient high-throughput sequencing platforms, massive biomolecular sequences have been extraordinarily published and made available [102], including entire genomes and proteomes. For example, the number of completely sequenced genomes stored in the Genome OnLine Database has already reached the impressive number of 1,994. As of today, there are more than 100 millions DNA sequences in GenBank, comprising 100 billions base pairs (Gbp); about 18.5 million protein sequences in the EBI UniProtKB/TrEMBL database, comprising 6 billions amino acids, and 78,304 known structures in the RCSB Protein Data Bank. The size of the entire human genome is in the order of 3 billion DNA base pairs, and only few methods can actually handle sequences of this length.

The analysis and processing of massive biological sequences, or strings, thus require great efforts, and are typically made by means of subsequence decompositions [23; 48; 55; 86]. Here patterns are models of information underlying the extracted subsequences, containing knowledge about the biological samples and their intrinsic regularities. In classification tasks, these features are employed to compute a similarity or distance measure between two or more sequences, and then to train a classifier [101]. Thereby a central role is played by the notion of *motif*, i.e., a pattern repeated several times in a set of sequences, most likely corresponding to some functional role or to a trace of the evolution [94].

When dealing with massive data analysis, the discovery and matching of such motifs can be prohibitive, with the number of motifs that can possibly escalate exponentially in the size of sequences [8]. The main underlying difficulty is that, in order to model mutations and other evolutionary mechanisms, motifs have to account for some degree of variability. For these reasons, motifs are often modeled to include indeterminate symbols, representing errors or mismatches (e.g., *don't care* symbols), sequence profiles, or classes of symbols representing allowed substitutions, e.g., `C[GR][ILMV]C` [33]. The major distinction is between deterministic and probabilistic motifs [13], where the former class represents patterns that deterministically appear in certain locations of the sequences under consideration, and will be the focus of our

work.

To handle the combinatorial explosion of motifs, a number of different techniques have been proposed over the past two decades [48; 57]. In general these techniques make a trade-off between the information captured and its size and ease of computation, such as happens with text/image compression [6; 7; 69; 71; 93]. A typical scenario is that of shrinking the search space according to the syntax and/or statistical distribution of motifs [4; 14; 55]. Our general insight is that the total length of functional motifs in a biological sequence, and not only their whole number, should be much smaller than the size of the sequence itself. Accordingly, our belief is that motifs shared by a number of sequences that cover vast parts of them, well characterize the biological function of those regions, and result in a collection of information whose dimension is linear in the size of the original sequences. In this thesis we address these issues, introducing new notions of meaningful motifs that can be employed for protein and genome analysis.

1.3 Main Results of the Proposed Research

Looking for this purpose, we developed two paradigms of motif discovery and filtering, namely *irredundant common motifs* and *underlying motifs*, which do not rely on expensive alignment techniques.

The first approach systematically detects the smallest class of "independent" motifs, or *irredundant*, that can describe all common motifs in two sequences. We say that a motif m is redundant if its list of occurrences can be deduced by a number of other common motifs. Accordingly, we say that a motif occurrence is covered by the occurrences of other common motifs, if these latter motifs are more specific and/or extend m to the left or to the right, appearing in the same site. Any redundant motif can thus be derived from the set of all *irredundant common motifs* without knowing the original sequences, and therefore is not informative for this measure. Moreover, we provide evidence that almost all approaches based on subsequence decompositions employ motifs, of various forms, that are redundant in our model, most likely misleading the analysis and classification of sequences. For example, let us define the information shared by two sequences as the number

of symbols that match together. In the popular k-mer framework, which captures all the shared subwords of fixed-length k between two sequences —i.e., motifs with no undecided symbols,— this corresponds to $O(kn)$ symbols overcounted because of overlapping motif occurrences, which leads to a major redundancy in case of massive test sets.

On the other hand, the information given by motifs of various types may include another kind of noise due to the repetition per se. In fact, some motifs may occur in the same region on the sequences analyzed, when only one of them is actually the functional motif for that site. In the second approach we thus consider a set of motifs given *a priori*, and we aim to select the most representative ones for each region of the sequences according to some priority rule. We call these motifs, *underlying motifs*. In this framework one can use the most varied binary relations in place of the priority. Here we have developed a simple and practical fingerprint rule, called *motif priority*, which is based on length, composition (in case of motifs with character classes), and the lexicographic order of occurrences for each motif. Inherently the resulting set of underlying motifs will be linear in the size of the original sequences. Furthermore, it can be shown that the most popular motif discovery tools discover motifs whose total length does not scale linearly with the size of sequences; therefore, a "space-conscious" approach of this type will allow us to manage the selected motifs easier, and will enhance their readability and interpretation. Our priority rule can finally rank all motifs and give in output an ordered list of the most relevant ones.

The paradigms described have been published in [27; 28; 29] and applied to the analysis and classification of protein sequences. In this context, we further show how to combine these approaches to allow the comparison of whole genomes.

In this thesis, we provide practical and efficient methods for solving three fundamental problems that are emerging in the field of computational molecular biology:

1. classification of protein sequences;
2. identification of subtle biological signals;
3. whole-genome phylogeny reconstruction for different species.

The first problem mainly concerns the identification of remote evolutionary relationships among proteins, when their sequences present low or sparse similarity and thus are difficult to digest for general methodologies based on alignments. In this case we employ the notion of irredundant common motifs, in the acceptation with don't cares (e.g., a motif of the type `DPC·IM··C`), and the statistics of these patterns to compute a discriminative learning function able to distinguish the major protein classes. In particular, this function exploits the power of support vector machines (SVMs), a tool that is currently emerging as one of the most common and practical means in science for classifying a set of comparable objects. For each pair of protein sequences we compute a similarity score based on the extracted patterns, and then employ the resulting scores as a kernel in the SVMs. Our method, called *Irredundant Class*, has proved to be valuable for the identification of remote protein homologies, successfully classifying protein sequences in a well-known benchmark.

Another important problem in molecular biology is the identification of subtle biological signals in a set of closely related sequences. This concerns the discovery and filtering of overrepresented *degenerate motifs* present in a set of evolutionarily-related sequences, in order to identify which motifs are affiliated with particular regulatory or functional sites. In our settings, a degenerate motif is a motif with character classes; for example, `[FY]D[IP][CU][AILMV][AGS]C` is one of the two functional motifs for the Nickel-dependent hydrogenases protein family. Degenerate motifs cannot be detected with a simple screening of the entire sequences, thus biologists resort to particular motif discovery tools, mostly based on sophisticated heuristics, capable of finding complex motifs. In general, a few state-of-the-art discovery tools are available for each biological problem, and the output of these methods is often extraordinarily huge and complicated due to the high degree of variability given by suitable character classes. In this context we propose a measure for simplifying the output of these powerful tools, associating at most one significant motif to each site of the sequences analyzed, thus improving the subsequent screening and editing by biologists. The general approach is based on the notion of underlying motifs in conjunction with that of motif priority. Experimental results evidence that we are able to identify functional

degenerate motifs in two large families of protein sequences, by filtering and ranking all motifs in output from a state-of-the-art tool in protein analysis.

Finally, we reconstruct the phylogeny of different species by comparing whole genomes. This is a task that has recently emerged with the availability of completely sequenced genomes. First tests have shown that the evolutionary information is also carried by the non-coding material [104]. In this work, we construct a novel distance function between entire genomes by extending a practical approach in this field, the Average Common Subword approach, and combining the paradigm of irredundant common motifs with that of underlying motifs.

Great efforts are required for comparing whole genomes, where we must take into account both the coding and non-coding regions, and pay attention to the myriad of repeats present in a sequence of this type. In this case we use the two proposed paradigms in the context of subwords, also called in the literature substring motifs, exact patterns, or contiguous subsequences; for example, the so-called TATA box `TATAAAA` is a common DNA subword found in the promoter region of genes in archaea and eukaryotes. This allows the use of efficient data structures, such as suffix trees and suffix arrays, which are able to digest sequences of large size, and then to compute the irredundant common subwords in a pairwise fashion. We can further prove that these particular subwords correspond to a measure closely related to the Kullback-Leibler divergence estimated between two genome sequences. This approach, alone, may overcount the same regions of a sequence, without emphasizing motifs conserved during evolution, which usually reside in different genome sites. Thus, we address these issues by applying the notion of underlying motifs to the irredundant common subwords. We call the subwords resulting from this pipeline, *unic subwords*. Experimental results on the reconstruction of phylogenies by means of unic subwords show very good performance with respect to other state-of-the-art approaches. In particular, we have successfully applied our method, called *Unic Subword Approach*, to the following species: viruses of the 2009 human pandemic Influenza A (H1N1); prokaryotes from both the Archaea and Bacteria domains; and unicellular eukaryotes of the genus Plasmodium, which are related to the severe disease of malaria.

1.4 Logical Organization of the Thesis

This thesis is presented in three chapters divided by topics. Each topic develops one of the three challenges mentioned above separately. For each chapter we introduce first, in a summary, the problem faced and the main results achieved. Then, we make an overview of the problem background, the state-of-the-art methods, and the related work. We subsequently design methods able to attack the problem, by specifying a theoretical framework; thus, we describe algorithms and data structures employed, and analyze the total computational complexity using the RAM model of computation. We finally present some experimental results and further analysis on protein and genome sequences.

The two main techniques of pattern analysis discussed above, namely irredundant common motifs and underlying motifs, will be introduced in chapters 2 and 3, respectively, along with the corresponding applications.

In particular, in Chapter 2 we characterize the remote homology of protein sequences; these proteins analyzed are related by similar tertiary structures and/or functions, but have low sequence similarity. Thus, we model this aspect with don't cares. In this case our method, the Irredundant Class, successfully classifies 4,352 protein sequences, accounting for 560,000 amino acids in total. Specifically, we assess the performance of the Irredundant Class on a challenging benchmark consisting of 54 different experiments based on these sequences, where many other researchers have faced in the last ten years [27; 28].

In Chapter 3, we compare, rank, and filter degenerate motifs, in order to simplify the output of modern motif discovery tools and identify subtle signals in functional or conserved sites from a set of biological sequences. Our method exploits the motif priority rule to achieve a total ranking of all degenerate motifs in input, and then selects a significant motif for each site through the filter given by underlying motifs. We perform experiments on two large families of protein sequences, amounting to 12,000 and 46,000 amino acids, respectively. Results show that we drastically reduce the number of degenerate motifs in output from a state-of-the-art tool in protein analysis, retaining and ranking in the top 5 positions the actual functional motifs

manually identified [29].

In Chapter 4, we finally present a distance function for whole genomes based on the unic subwords. Experiments focus on the reconstruction of phylogenies for different organisms, whose size ranges from 13 kbp for viral genomes, to 24 Mbp for the eukaryotic genomes of Plasmodium; moreover, the tested prokaryotic genomes amount on average to 2,700 Mbp. The reference taxonomy is computed using standard methods taken from the literature for viruses and prokaryotes, while we use a well-established phylogeny for Plasmodium. Results indicate that we are able to identify the major clusters found in these taxonomies.

In summary, the proposed approaches have been successfully applied to the analysis and processing of massive biological sequences for solving three fundamental problems in molecular biology. The logical organization of the thesis reflects these three issues involved.

Chapter 2

The Irredundant Class Method for Remote Protein Homology Detection

The automatic classification of protein sequences into families, or homology investigation, is of great help for the functional prediction and annotation of new proteins. In this chapter we present a novel computational method, called Irredundant Class, addressing the remote protein homology detection problem. This problem is to recognize homology in case of low sequence similarity, by relating protein domains that share a common evolutionary origin and perhaps similar tertiary structures and biological functions. For instance, actin, the ATPase domain of the heat shock protein, and hexokinase, an enzyme that acts on glucose, are a class of functionally related proteins with low sequence similarity. A knowledge of these relationships without resorting to tertiary structure analysis, often infeasible, is crucial to our understanding of life.

The remote homology detection is a hard computational problem with no existing approach that works well in all cases [18]. The best performing methods that partially solve this problem are discriminative string kernels. These methods compute a similarity function between pairs of proteins based on their subsequence composition, and then combine the resulting scores in order to distinguish the major classes. We provide evidence that almost all string kernels are based on patterns that are not independent, and therefore the associated similarity scores are obtained using a set of redundant fea-

tures, overestimating the similarity between proteins. To specifically address this issue we introduce the class of *irredundant common motifs with don't cares*. Loosely speaking, the set of irredundant common motifs is the smallest class of independent patterns that can describe all shared patterns with mismatches in a pair of sequences. Thus, we present a classification method based on the statistics of these patterns. Results on benchmark data show that the Irredundant Class outperforms most of the string kernels previously proposed, and achieves results as good as the current state-of-the-art method, but using the same pairwise information only once.

2.1 Background

2.1.1 Related Work

The protein classification problem can also be treated as a binary string classification problem. Historically this problem has been studied, for quite some time, in the field of text documents classification [77]. Unfortunately most of the proposed approaches, developed for a different kind of strings, fail when applied to biological sequences. The main reason of this failure is the different nature of biological sequences, particularly rich of regularities known as patterns that are difficult to digest for a general purpose application.

A number of methods have been studied for the classification of evolutionarily-related proteins based on primary structure. The main distinction is between generative methods versus discriminative methods. The former class includes methods such as protein family profiles [45], hidden Markov models (HMMs) [16; 60; 63], and PSI-BLAST [3]. These methods derive a model for a set of proteins and then check whether a candidate protein fits the model or not. Unlike generative methods, discriminative methods such as [51; 54; 64; 71; 72; 74; 92; 98] focus on finding which sequences can describe a set of proteins despite of another set.

Recent results [71; 74] suggest that the best-performing methods are discriminative string kernels. These methods use kernel functions based on common motifs between pairs of protein sequences to train a support vector machine (SVM) [32; 114]. The string kernel extracts information from se-

quences and computes either a feature vector for each sequence or directly a kernel matrix with scores between pairs of sequences. In this case, sequences are seen as a set of labeled examples —positive if they are in a particular protein family, and negative otherwise— and an SVM attempts to learn a decision boundary between the different classes.

The first string kernel, called Fisher's kernel [54], uses an HMM to provide the necessary means of converting proteins into fixed-length vectors. The vector summarizes how different the given sequence is from a typical member of the given protein family. In contrast, in the Pairwise empirical kernel [74] the feature vector corresponding to a protein sequence is formed by all E-values, given by the Smith-Waterman algorithm [109], computed on the sequence analyzed and each of the training sequences of a particular experiment.

Different methods, like the Spectrum and the Mismatch kernels [71; 72], use as protein features the set of all possible subwords of amino acids of fixed length (k-mers). If two sequences contain many of the same k-length contiguous subsequences, their inner product under the k-Spectrum kernel will be large. Equivalently, the Mismatch kernel computes a large inner product between two sequences if these sequences contain many k-length contiguous subsequences that differ by at most e mismatches.

More recently, the Local Alignment method [98] tried to mimic the behavior of the Smith-Waterman algorithm to build a family of valid kernels. Following the work of [49] they defined a kernel that mimics the detection of all local alignments between two sequences by convolving simpler kernels. Another recent approach, the Word Correlation Matrices method [75], defines the kernel by average pairwise subword similarity between two sequences, similarly to the Spectrum kernel. The consequent analysis of discriminative subwords allows also for the identification of characteristic regions in biological sequences.

Other methods, such as the I-Sites [51], encode into feature vectors information related both to three-dimensional structure properties and sequence similarities of proteins. Conversely, the eMOTIF database approach [18] defines a kernel function in terms of the occurrences of sequence motifs previously stored in a database, and typically extracted using popular algorithms

on reference sequences [52; 83].

Finally, Profile-based Mismatch methods [64] use probabilistic profiles, such as those produced by the PSI-BLAST algorithm, to define kernels based on position-dependent mutation neighborhoods of k-mers with mismatches, in a similar way to the original Mismatch kernel. Variants of this, as the Profile-based Direct methods [92], build a kernel function combining sequence profiles obtained with different approaches for determining the similarity between pairs of protein sequences. Note that the latter methods make an extensive use of hyperparameters, increasing the risk of overfitting the classification problem when no dedicated validation dataset is used.

2.1.2 A General Insight

We selected for comparison with our method some of the algorithms presented above, in particular those with state-of-the-art performance on the classification of proteins which seem somehow more reliable: Fisher, Pairwise, Spectrum, Mismatch, Local Alignment (version "eig"), and Word Correlation Matrices. In general, all pattern-based methods operate two distinct steps: first extract and process common motifs from a pair of sequences, then use this set of patterns as features to build an automatic classification tool based on SVMs; and so does the method proposed here. As we will show in the next sections, almost all string kernels are based on patterns that are not independent, namely patterns with occurrences that are related each other. Any score built using a set of related patterns is in practice based on redundant features, and therefore it can potentially overestimate the similarity score.

In this chapter we want to stress the idea that if the learning process has to deal with a set of *redundant* features, this might mislead the classification of sequences. The goal is somehow similar to the feature selection problem, but, in the case of pattern-based methods for classification contexts, the class of *irredundant* common motifs is specifically designed to address the issue of a repeated information. Our conjecture is that a set of *irredundant* common motifs, and consequently a set of independent features, can facilitate the automatic learning and classification of sequences.

2.2 Materials and Methods

2.2.1 *Irredundant Class*

The method is based on the extraction and filtering of patterns with mismatches shared by two sequences, also called *common motifs*, using the mathematical notion of *irredundancy*. This notion was first studied by [43; 85; 87] for the case of repeated patterns on a single sequence, and thereafter by [9; 89]. We extended the notion of irredundancy to the case of two sequences, in order to avoid the overcount of common motifs that "cover" the same region of the sequences multiple times. Indeed, one can easily show that there are lots of protein sequences that share an unusually large number of common motifs without conveying extra information about the input (see, for example, Table 2.7 presented later in the chapter).

In the following we present the class of irredundant common motifs, along with some properties. Let s_1 and s_2 be two sequences, or strings, of m and n characters respectively, over an alphabet Σ. A character from Σ, say σ, is called a *solid* symbol, while a *don't care* symbol '$\cdot$', i.e., a mismatch, *equals* and represents any character on Σ. When dealing with protein sequences, Σ corresponds to the 20 amino acids.

The set of all strings over an alphabet $\mathcal{A}$ is denoted by $\mathcal{A}^*$. The length of a string x is defined as the number of its symbols and denoted by $|x|$. We define $x[i]$ and $x[i, j]$ to be, respectively, the i-th symbol of x and the subsequence given by the $j - i + 1$ consecutive characters of x starting from position i.

A *pattern* p is a string over $\Sigma\,(\Sigma \cup \{\cdot\})^*\,\Sigma$, thus having at least two solid characters: the first and the last character. A set of locations $[i, j]$ of s_1 is an *occurrence* of p if $s_1[i, j] = p$; we usually denote an occurrence of p by its first location, in this case i. For instance, $p =$ `D·DG·G·I· · ·E` is a pattern that occurs at the locations 1 and 15 of the sequence $s_1 =$ `DADGGGDISTKETVDEDGSGTIDFEE`. A *common motif* is a pattern that occurs in both s_1 and s_2. We finally call a *suffix* of s_1 any subsequence of the type $s_1[i, m]$, with $1 \leq i \leq m$.

Definition 2.1. *(Common motif, Location list) Let s_1 and s_2 be two se-*

quences on Σ*. A pattern* p *on* $\Sigma(\Sigma \cup \{\cdot\})^*\Sigma$ *is a common motif with location list* $\mathcal{L}_p = (l_1, l_2, \ldots, l_q)$ *if all of the following hold:*

(i) p *occurs both on* s_1 *and* s_2*,*

(ii) p *occurs in all the locations* $l \in \mathcal{L}_p$*, and*

(iii) there exists no location $l \notin \mathcal{L}_p$ *such that* p *occurs at* l *belonging either to* s_1 *or to* s_2 *(i.e., the location list is of maximal size).*

Clearly, a common motif occurs at least twice, one per sequence. Extending the notation of location list we can then denote by $(\mathcal{L}_p+i)$, $0 \leq i \leq |p|-1$, all the locations in $\mathcal{L}_p$ shifted to the right by i positions.

To give an idea of the notion of irredundancy applied to two sequences s_1 and s_2, let us consider $s_1 =$ `ABABABAB` and $s_2 =$ `BABABABA`, and the two patterns $p_1 =$ `ABABABA` and $p_2 =$ `ABABA` that are contained in both the sequences. We can note that the existence of p_1 in both s_1 and s_2 affects the existence of p_2 in all locations in which p_2 appears. By simply deleting the last `BA` from p_1 or right shifting twice p_2 along p_1 we can obtain p_2. Loosely speaking, the two common motifs p_1 and p_2 are not independent or, equivalently, they are not irredundant. Intuitively, a common motif is irredundant if it cannot be deduced, along with its location list, by some other patterns. Consequently, any redundant common motif can be derived from the set of all irredundant common motifs without knowing the original sequences, thus it is not informative for this measure. We want to discard all redundant common motifs as non-informative for the learning process in our extent.

Definition 2.2. *(*$p_1 \preceq p_2$*) For characters* σ_1 *and* σ_2 *we write that* $\sigma_1 \preceq \sigma_2$ *if and only if* σ_1 *is a don't care or* $\sigma_1 = \sigma_2$*. Given two distinct patterns* p_1 *and* p_2*, with* $|p_1| \leq |p_2|$*,* $p_1 \preceq p_2$ *holds if* $p_1[j] \preceq p_2[j+d]$ *for all* $1 \leq j \leq |p_1|$*, with such an offset* $0 \leq d \leq |p_2| - |p_1|$*.*

We also say in this case that p_1 is a *subpattern* of p_2 (or p_1 occurs in p_2), and that p_2 implies or extends p_1.

Definition 2.3. *(Maximal common motif) Let* $\mathcal{P}_{s_1,s_2} = \{p_1, p_2, \ldots, p_k\}$ *be the set of common motifs of the sequences* s_1 *and* s_2*. A common motif* $p_i \in$

$\mathcal{P}_{s_1,s_2}$ is maximal in composition if and only if there exists no common motif $p_l \in \mathcal{P}_{s_1,s_2}$, $l \neq i$, such that $p_i \preceq p_l$ and $\mathcal{L}_{p_i} = \mathcal{L}_{p_l}$. A common motif p_i maximal in composition is also maximal in length if and only if there exists no common motif $p_j \in \mathcal{P}_{s_1,s_2}$, $j \neq i$, such that $p_i \preceq p_j$ and $|\mathcal{L}_{p_i}| = |\mathcal{L}_{p_j}|$. A maximal common motif is a pattern that is maximal both in composition and length.

Requiring maximality in composition and length limits the number of common motifs that may be usefully extracted and accounted for in two sequences. However, the notion of maximality alone does not suffice to bound the number of such patterns. It can be shown that there are sequences sharing an unusually large number of maximal common motifs without conveying extra information about the input. To this end, we want to investigate on a more restrictive notion of informative patterns, introducing first the following definition of pattern occurrence coverage:

Definition 2.4. *(Occurrence coverage) Given two patterns p_1 and p_2 of q and $r \geq q$ characters, respectively, we say that the occurrence $[i, q+i-1]$ of p_1 on s_1 is covered by p_2 if and only if*

(1) $p_1 \preceq p_2$, with offset d, and

(2) p_2 occurs at $i-d$ or, equivalently, $s_1[i-d, i-d+r-1] = p_2$.

For instance, the common motif $p_1 =$ `ABABA` with location list $\mathcal{L}_{p_1} = (1_{s_1}, 3_{s_1}, 2_{s_2}, 4_{s_2})$ over $s_1 =$ `ABABABAB` and $s_2 =$ `BABABABA`, where j_{s_h} denotes the occurrence at location j in the sequence s_h, is covered at position 3_{s_1} by $p_2 = abababa$ with $\mathcal{L}_{p_2} = (1_{s_1}, 2_{s_2})$ and $i = 2$. Note that $\mathcal{L}_{p_2} \subseteq \mathcal{L}_{p_1}$ because p_1 is a subpattern of p_2 obtained by deleting the last *ba* from p_2 (i.e., the shift integer i is 0), and that $(\mathcal{L}_{p_2} + 2) \subseteq \mathcal{L}_{p_1}$ because p_1 occurs also at location 2 in p_2. Another example with don't cares is the following, $p_3 =$ `A·A·A` with $\mathcal{L}_{p_3} = (2_{s_3}, 4_{s_3}, 2_{s_4}, 4_{s_4})$ over $s_3 =$ `AABABABAB` and $s_4 =$ `BABACACAC` is covered at all the positions by $p_4 =$ `ABA·A·A` with $\mathcal{L}_{p_4} = (2_{s_3}, 2_{s_4})$.

In other words, property (1) of Definition 2.4 says that p_2 has to extend p_1 in composition (i.e., p_2 would be more specific than p_1) and/or in length, and (2) indicates that p_2 must occur in the same region of p_1. We give a

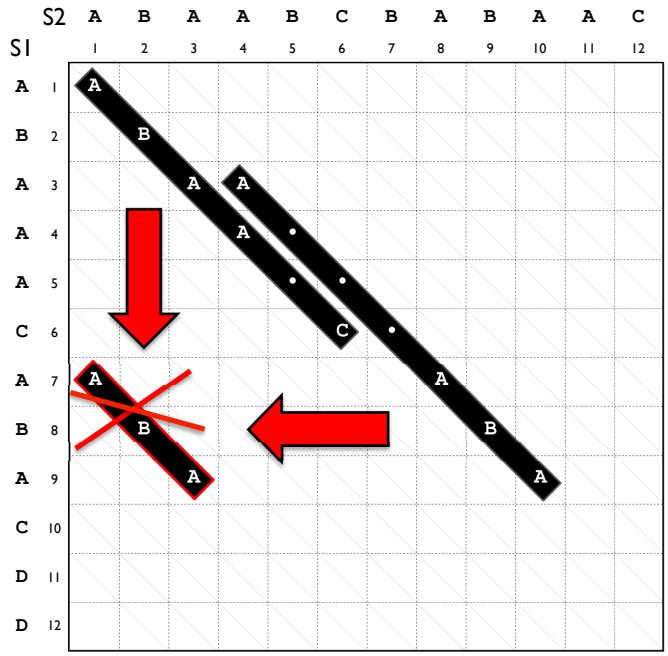


FIGURE 2.1. *Example of pattern occurrence coverage on the sequences* $s_1 =$ `ABAAA-CABACDD` *and* $s_2 =$ `ABAABCBABAAC` *of length 12. The occurrences of the meet* `ABA` *at* $\{1, 7\}$ *in* s_1 *and at* $\{1, 8\}$ *in* s_2 *are all covered by the meets* `ABAA·C` *and* `A···ABA`.

graphical example of coverage in Figure 2.1. In this case, we can easily see that the occurrences of the common motif `ABA` at 7 in s_1 and at 1 in s_2 are covered, respectively, by the common motifs `ABAA·C` and `A···ABA`.

Now, we can say that a maximal common motif p is *irredundant* if p and the list $\mathcal{L}_p$ of its occurrences is not covered by other maximal common motifs, i.e., $\mathcal{L}_p$ cannot be deduced by the union of a number of lists of subpatterns of other maximal common motifs. Conversely, we call a common motif p *redundant* if p (along with its location list $\mathcal{L}_p$) can be deduced from the other common motifs without knowing the input sequences. More formally:

Definition 2.5. *(Irredundant/Redundant common motif) A common motif* p *is* irredundant *if and only if at least an occurrence of* p *on* s_1 *or* s_2 *is not covered by other common motifs. Otherwise, we say that* p *is* redundant.

Since the redundant common motifs bring no extra information about the two sequences, the set of independent patterns is precisely the class of

irredundant common motifs. For instance, let us consider the sequences $s_1 =$ ABABABAB and $s_2 =$ BABABABA of length 8. Then, the list of all irredundant common motifs is the following: $p_1 =$ ABABABA with $\mathcal{L}_{p_1} = (1_{s_1}, 2_{s_2})$, $p_2 =$ BABABAB with $\mathcal{L}_{p_2} = (2_{s_1}, 1_{s_2})$. The other redundant maximal common motifs are: $p_3 =$ ABABAB with $\mathcal{L}_{p_3} = (1_{s_1}, 3_{s_1}, 2_{s_2})$, $p_4 =$ BABABA with $\mathcal{L}_{p_4} = (2_{s_1}, 1_{s_2}, 3_{s_2})$, $p_5 =$ ABABA with $\mathcal{L}_{p_5} = (1_{s_1}, 3_{s_1}, 2_{s_2}, 4_{s_2})$, $p_6 =$ BABAB with $\mathcal{L}_{p_6} = (2_{s_1}, 4_{s_1}, 1_{s_2}, 3_{s_2})$, $p_7 =$ ABAB with $\mathcal{L}_{p_7} = (1_{s_1}, 3_{s_1}, 5_{s_1}, 2_{s_2}, 4_{s_2})$, $p_8 =$ BABA with $\mathcal{L}_{p_8} = (2_{s_1}, 4_{s_1}, 1_{s_2}, 3_{s_2}, 5_{s_2})$, $p_9 =$ ABA with $\mathcal{L}_{p_9} = (1_{s_1}, 3_{s_1}, 5_{s_1}, 2_{s_2}, 4_{s_2}, 6_{s_2})$, $p_{10} =$ BAB with $\mathcal{L}_{p_{10}} = (2_{s_1}, 4_{s_1}, 6_{s_1}, 1_{s_2}, 3_{s_2}, 5_{s_2})$, $p_{11} =$ AB with $\mathcal{L}_{p_{11}} = (1_{s_1}, 3_{s_1}, 5_{s_1}, 7_{s_1}, 2_{s_2}, 4_{s_2}, 6_{s_2})$, $p_{12} =$ BA with $\mathcal{L}_{p_{12}} = (2_{s_1}, 4_{s_1}, 6_{s_1}, 1_{s_2}, 3_{s_2}, 5_{s_2}, 7_{s_2})$.

It is easy to check that p_1 and p_2 cannot be deduced by other common motifs, whereas p_5 along with its location list can be simply deduced by p_1, and p_7 can be derived from the union of the occurrence lists of some other common motifs, that in practice are subpatterns of p_1 and p_2. We want to emphasize that if the two sequences are identical there is only one irredundant common motif, the sequence itself. This difference with the single sequence approach is due to a peculiarity of the original notion, in which the sequence itself is not considered as a valid pattern.

Remark 2.1. *Let s_1 and s_2 be two sequences, and p_1 and p_2 be two common motifs with $p_1 \preceq p_2$. Then, by definition p_2 must cover at least an occurrence of p_1 per sequence.*

Definition 2.6. *(Meet) The* meet *of two subwords of s_1 and s_2 is obtained by matching the characters corresponding to the same positions, inserting a don't care in case of mismatch, and thereafter deleting all leading and trailing don't cares. Furthermore, every meet is also a common motif if it has at least two solid characters.*

For instance, the consensus of the sequences AAAAAB and BAAAAA is ·AAAA·, while their meet is AAAA. Note that a meet is a common motif between two sequences.

Consider now the sequence $s_1 = \sigma_1\sigma_2 \ldots \sigma_n$ of n characters over Σ. Then, we use $suf_j^{s_1}$ to explicitly denote the suffix $\sigma_j\sigma_{j+1} \ldots \sigma_n$ of s_1 (i.e., $s_1[j, n]$), $s^{(j)}$ the sequence where the location j appears, and $\mathcal{L}_p^{s_1}$ the location list of a

common motif p on s_1. Clearly $suf_1^{s_1} = s_1$, $s^{(1_{s_1})} = s_1$, and $\mathcal{L}_p = \mathcal{L}_p^{s_1} \bigcup \mathcal{L}_p^{s_2}$. Moreover, $j \in \mathcal{L}_p$ if and only if $j \in \mathcal{L}_p^{s^{(j)}}$. In the following we will briefly present the most important properties of the irredundant common motifs. Those properties are specular with respect to the single sequence approach, as shown in [10].

Lemma 2.1. *Let p be a common motif of s_1 and s_2. Then, p is irredundant if and only if there exists $j \in \mathcal{L}_p^{s^{(j)}}$ such that the meet of $s^{(\min\{j,k\})}$ and $suf^{s^{(\max\{j,k\})}}_{\max\{j,k\}-\min\{j,k\}+1}$ is p for all $k \in \mathcal{L}_p^{s_h}$, where s_h is the other sequence with respect to $s^{(j)}$. We call the occurrences that satisfy this property,* exposed.

Proof. We recall that a common motif p is irredundant if and only if it has occurrences in s_1 or in s_2 that are not covered by other common motifs. We will show that these occurrences are precisely the exposed occurrences of p. Let us define the meet p that correlates a pair of locations (j, k) of $s^{(j)}$ and $s^{(k)}$, with $s^{(j)} \neq s^{(k)}$, as the intersection of the two entire sequences such that j and k overlap each other, and p occurs within j and k —i.e., p occurs at these locations less possibly an offset or, equivalently, $j, k \in (\mathcal{L}_p + i)$ with $0 \leq i \leq |p|-1$. Certainly this corresponds to the meet of $s^{(\min\{j,k\})}$ with $suf^{s^{(\max\{j,k\})}}_{\max\{j,k\}-\min\{j,k\}+1}$, if it occurs within j and k. Furthermore, it is easy to see that, in this case, a meet of length r correlates r location pairs with consecutive values. For instance, in Figure 2.1 the pattern `A`$\cdots$`ABA`, in black, is the meet that correlates the pairs $(3_{s_1}, 4_{s_2})$,$(4_{s_1}, 5_{s_2})$,$(5_{s_1}, 6_{s_2})$,$(6_{s_1}, 7_{s_2})$,$(7_{s_1}, 8_{s_2})$,$(8_{s_1}, 9_{s_2})$, $(9_{s_1}, 10_{s_2})$.

Now, let us first show that every irredundant common motif has at least an exposed occurrence. If p is irredundant, then it has at least an occurrence that is not covered by other common motifs, say j. Therefore p must result from the meet correlating the pair of locations (j, k) of the sequences, for all possible $k \in \mathcal{L}_p^{s_h}$, with $s_h \neq s^{(j)}$. Otherwise, following the definition of meet, the occurrence j of p would be covered by the common motifs that result from these meets, contradicting our assumption. It follows that j is an exposed occurrence for the pattern p.

Conversely, if j is an exposed occurrence of a common motif p, there can be no other common motif p' that covers j, otherwise, for Definition 2.6 and Remark 2.1, p' would result in the meet correlating the locations (j, k) of the sequences, for some $k \in \mathcal{L}_p^{s_h}$ (with $s_h \neq s^{(j)}$). We can conclude that every

irredundant common motif must have at least an occurrence j, in any of the two sequences, that satisfies the second part of the lemma, and vice versa. □

To better clarify the meaning of this lemma, we refer the reader to the general example (Example 2.2.2) that follows the algorithm in the next subsection.

Theorem 2.1. *Every irredundant common motif of s_1 and s_2 is the meet of a sequence with a suffix of the other one.*

Proof. In Lemma 2.1 we showed that an irredundant common motif must appear at least in the meet resulting from an intersection of the two entire sequences. This corresponds to the meet of one entire sequence with a suffix of the other sequence. □

For instance, in Figure 2.1 the common motif `ABA` is the meet of s_2 with $suf_7^{s_1}$ of s_1 (i.e., the suffix $s_1[7, 12]$). However `ABA` turns out to be in any case a redundant common motif, and thus we need a more sophisticated algorithm to discover the whole class of irredundant common motifs, or *Irredundant Class* $\mathcal{I}_{s_1,s_2}$, as we call this set. In this regard, we will show how to exploit the power of Lemma 2.1 along with Algorithm 2.1 presented in the next subsection.

An immediate consequence of Theorem 2.1 is a linear bound for the cardinality of the set of irredundant common motifs. Thus:

Theorem 2.2. *The number of irredundant common motifs over two sequences s_1 and s_2 of length, respectively, m and n is $O(m + n)$.*

Proof. As of Theorem 2.1, the maximum number of meets between a sequence with the suffixes of the other one is limited in number by the length of s_1 and s_2. These common motifs, necessarily of length greater than 1, are at most $m + n - 3$. □

With its underlying convolutory structure, Lemma 2.1 suggests an immediate way for the extraction of irredundant common motifs from sequences and arrays, using available pattern matching with or without FFT [10]. For

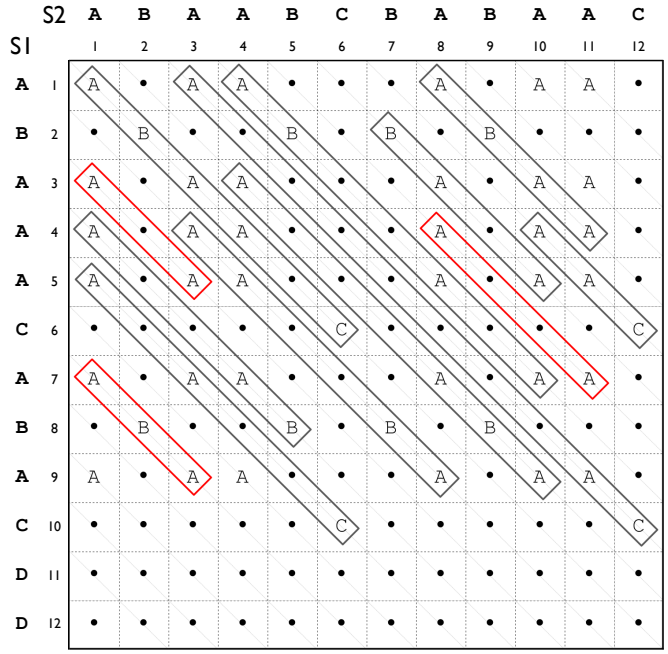


FIGURE 2.2. *Irredundant common motifs, in black, for the sequences* $s_1 =$ `ABAAA-CABACDD` *and* $s_2 =$ `ABAABCBABAAC` *of length 12. In red are highlighted the redundant common motifs (*`ABA`*,* `A·A`*,* `A··A`*) among all the meets between a sequence and a suffix of the other sequence.*

example, Figure 2.2 displays in black the irredundant common motifs for two sequences, among all the considered meets between a sequence and the suffixes of the other sequence.

2.2.2 The Proposed Algorithm

The discovery of all the irredundant common motifs $\mathcal{I}_{s_1,s_2}$ over two sequences s_1 and s_2 is derived from Lemma 2.1.

Follows the complete description of the algorithm, where the input are two sequences s_1 and s_2 over Σ, with $|s_1| = m$ and $|s_2| = n$, and the output is the Irredundant Class $\mathcal{I}_{s_1,s_2}$.

Algorithm 2.1.

1. Compute the $m+n-3$ meets between s_1/s_2 and a suffix of the other sequence; then discard patterns of length < 2.
2. For each meet p:
3. (a) for each occurrence j of p found in step 1, called exposed occurrence, increment a counter relative to p ($I_1[j]$ or $I_2[j]$) depending on the sequence in which j appears;
4. (b) perform a string search over s_1 and s_2 to find the number of occurrences of p, called respectively q_1 and q_2;
5. (c) check if the meet p is irredundant by finding at least an exposed occurrence j of p in $s^{(j)}$ that has a counter value equal to the number of occurrences of p in the other sequence (with respect to $s^{(j)}$). Equivalently, find if there exists an occurrence j in s_1 such that $I_1[j] = q_2$, or an occurrence k in s_2 such that $I_2[k] = q_1$.

The algorithm complexity is dominated by the most computationally intensive operation, step 4, which is the global searching of all the occurrences of patterns in the sequences. Therefore, we can extract the Irredundant Class in time $O(z^2 \log z \log |\Sigma|)$, where $z = m + n$, making use of the FFT in the step of searching for occurrences of the $m + n - 3$ meets described above.[1]

Example of Irredundant Class Computation

Consider the sequences $s_1 =$ `AABABABAB` and $s_2 =$ `BABACACAC` of length 9. One of the meets computed by Algorithm 2.1 is the meet of $s^{(\min\{1_{s_1}, 3_{s_2}\})}$ and $suf^{s^{(\max\{1_{s_1}, 3_{s_2}\})}}_{\max\{1_{s_1}, 3_{s_2}\} - \min\{1_{s_1}, 3_{s_2}\} + 1}$ that is equivalent to compute the meet of $s^{(1_{s_1})}$ and $suf_3^{s^{(3_{s_2})}}$. Finally, it can be simply expressed as the meet of s_1 and $suf_3^{s_2}$, which is actually $p =$ `A·A·A` (see Table 2.1).

The only exposed occurrences of the common motif p are 2_{s_1} and 4_{s_2}, given by the meet correlating the pair $(2_{s_1}, 4_{s_2})$; thus $I_1[2] = 1$ and $I_2[4] = 1$. Accordingly, Table 2.2 shows the counters I_1 and I_2 for p at each location of s_1 and s_2, respectively.

[1] This step is described in detail in [40].

TABLE 2.1. EXAMPLE OF A MEET

position j	1	2	3	4	5	6	7	8	9		
$s_2[j]$	B	A	B	A	C	A	C	A	C		
$s_1[j]$			A	A	B	A	B	A	B	A	B
position j			1	2	3	4	5	6	7	8	9
p				A	·	A	·	A			

Meet between s_1 and a suffix of s_2, $suf_3^{s_2}$, where s_1 = AABABABAB and s_2 = BABACACAC.

TABLE 2.2. EXAMPLE OF COUNTERS FOR A MEET

position j	1	2	3	4	5	6	7	8	9
$I_1[j]$	0	1	0	0	0	0	0	0	0

position j	1	2	3	4	5	6	7	8	9
$I_2[j]$	0	0	0	1	0	0	0	0	0

Counters I_1 and I_2 for the common motif p = A·A·A, that is the meet between s_1 and $suf_3^{s_2}$, for each position of the sequences s_1 = AABABABAB and s_2 = BABACACAC.

We note that $z_1 = \max_{1 \leq j \leq |s_1|=9}\{I_1[j]\} = 1$ and that $z_2 = \max_{1 \leq j \leq |s_2|=9}\{I_2[j]\} = 1$. Then step 4 performs a string search of p over s_1 and s_2. We obtain that $\mathcal{L}_p^{s_1}$ is $(2_{s_1}, 4_{s_1})$ with cardinality $q_1 = 2$, and that $\mathcal{L}_p^{s_2}$ is $(2_{s_2}, 4_{s_2})$ with cardinality $q_2 = 2$. Since $z_1 < q_2$ and $z_2 < q_1$ we can conclude by Lemma 2.1 that p is redundant.

2.2.3 Scoring the Irredundant Class

Once acquired the Irredundant Class $\mathcal{I}_{s_1,s_2}$ of s_1 and s_2, we score this set of patterns exploiting their frequencies and some properties of the amino acid composition. Here we report the general form of the scoring function:

$$Score(s_1, s_2) = \ln\left(\sum_{p \in \mathcal{I}_{s_1,s_2}} \frac{F_p}{E[F_p]}\right),$$

where F_p is defined as the number of occurrences of the common motif p in s_1 and s_2, and $E[F_p]$ is the expected value of F_p.

To compute the expected value of F_p we assume that the sequences are drawn from an independent and identically distributed process (i.i.d. process). The probability of a pattern p is simply the product of the probabilities of its symbols $a_i \in p$. We extract the probability of a symbol in Σ from the BLOSUM62 substitution matrix [50], whereas we fix the probability of a don't care to 1. Since we have assumed that the sequences come from an i.i.d. process, the expected number of occurrences of p in s_1 and s_2 is:

$$E[F_p] = (m + n - 2(|p| - 1)) \times \prod_{a_i \in p} P(a_i),$$

where m, n, and $|p|$ are, respectively, the lengths of the two sequences and of the pattern p. Given a set of N sequences, the input for the SVM learning process is the resulting matrix of scores, i.e., $Score(s_i, s_j)$, with $1 \leq i, j \leq N$.

Unfortunately the Irredundant Class, the name as we will also call our general method in the rest of the chapter, seems to lack the positive-definiteness property, and therefore it must be treated as an indefinite kernel. In particular, following the work of [36] for indefinite kernels applied to SVMs, we have that the Irredundant Class is in the case of weak non-positivity, and thus we only need to set the SVM optimizer to possibly stop after a maximum number of iterations. Despite these manageable problems, we successfully applied the matrix of scores as a kernel matrix in the SVMs, and we retain for future work the task of bridging the theoretical gap in the non-positivity of the learning function.

2.3 Why Resort to the Irredundant Class?

The exhaustive homology detection in protein families and superfamilies leads to prohibitive computational methods; on the other hand, a low-complexity detection, for example using k-mers, would only consider a low-significant set of possible similarities and conserved amino acids, often overcounting the same information. These issues might be solved using an alternative method based on the irredundant common motifs with don't cares. Moreover, the

automatic filter given by the notion of "non-redundancy," or *irredundancy*, ensures us to select just those informative patterns, for this extent, that characterize the similarity of a pair of sequences.

We selected five algorithms of pairwise string similarity detection, used by state-of-the-art methods in protein classification, for the comparison with our approach: Spectrum, Mismatch, Word Correlation (the core of Word Correlation Matrices), Local Alignment (i.e., the distance function given by all local alignments), and Smith-Waterman (the core of Pairwise).

2.3.1 A Characterization of State-of-the-Art Pairwise String Algorithms

In the following we briefly explain the meaning of the selected algorithms on a pair of sequences s_1 and s_2, and then in the next subsection we estimate the redundancy, or *information overcount*, for each algorithm. Note that every algorithm computes a specific score for each extracted pattern, and then a global score is assigned to a pair of sequences using these pattern-specific scores.

In *Spectrum (k)* we count the number of occurrences for all the shared subwords of length k on Σ in s_1 and s_2. In *Mismatch* (k, e) we count the number of occurrences for all the shared strings of length k on Σ in s_1 and s_2, and then we add each value to the other k-mers of which the *meet* has at most e don't cares. In *Word Correlation (k)* we compute a similarity score between all the k-mers of s_1 versus all the k-mers of s_2, and this is like to consider all the meets on $\Sigma \cup \{.\}$ of k-length substrings of s_1 with k-length substrings of s_2. In *Local Alignment* we consider the global alignments between all pairs of substrings of s_1 and s_2 (given a scheme of scores for matches, substitutions, insertions, and deletions), that are all possible local alignments. Similar to Local Alignment, in *Smith-Waterman* we take the best global alignment between all pairs of substrings of s_1 and s_2. In *Irredundant Class* we consider all possible shared patterns on $\Sigma \cup \{.\}$ in s_1 and s_2, and then we avoid the contribution to be "overcounted" using the mathematical notion of *irredundancy* and selecting up to $m + n - 3$ patterns among the meets between all suffixes of s_1 and s_2.

2.3.2 Information Overcount: from a Theoretical Perspective

For each method we can now identify two characteristic phases: *(1) pattern extraction* and *(2) pattern processing.* We can think the output of phase (2) as a vector of pattern-specific scores, where each column represents just the score related to a single pattern.

For example, for Mismatch (1) is the process of finding k-mers in the two sequences s_1 and s_2, while (2) is the multiplication of the respective number of occurrences of these k-mers in s_1 and s_2, where the number of occurrences of each k-mer is the number of times it appears with up to e errors. In this case the output of phase (2) will be the vector of values resulting from the multiplications, and each column will represent a single k-mer. For Spectrum (1) is the same as for Mismatch, but (2) is only the multiplication of the shared k-mer occurrences without any other preliminary process, and thus without error parameters. For Word Correlation, in (1) we individually find the k-mers of s_1 and s_2, and in (2) we compute a similarity score between all the possible pairs of these k-mers (one k-mer of s_1 versus one of s_2). For Local Alignment (1) is represented by the extraction of all the substrings of the two sequences, while (2) is the global alignment of all the possible pairs of these substrings. For Smith-Waterman (1) and (2) are the same as for Local Alignment, but in (2) we have also a *max* operation between all the computed values on the possible global alignments. For Irredundant Class (1) is the extraction of all suffixes of s_1 and s_2, while (2) is the set of operations in which we compute the meets between a sequence and a suffix of the other one, and then we filter out the redundant ones.

Here we consider the information overcount as the number of outputs from phase (2) obtained taking into account the same pair of characters of s_1 and s_2 more than once:

Definition 2.7. *The* information overcount *is the number of vector components output of phase (2) in which the same pair of characters, one from s_1 and one from s_2, contributes more than once.*

Each output from phase (2), or component of the resulting vector, is intended as the score obtained comparing some pairs of single characters. For instance, after phase (2) of Spectrum we have a vector of values where

each column represents the multiplication of the numbers of occurrences of a specific k-mer found in s_1 and s_2. These k-mers are overlapped in the two sequences by construction, and each component of the resulting vector represents at least k positions of each sequence. Therefore we use an information about the comparison of a shared position between s_1 and s_2 in more than one k-mer, and thus we store this information in more than one column of the final vector, resulting in an information overcount. We call the model that considers the information overcount as the *Information Overcount Model*.

Table 2.3 shows a comparison of the algorithms based on the Information Overcount Model, where we fixed a priori $m \geq n$. The computation of these values is quite simple. For Irredundant Class, the meets between a sequence with all suffixes of the other sequence can be computed in a $m \times n$ grid, where each item represents the comparison of two different characters and each meet is a different diagonal of items from the top-left to the bottom-right part of the grid. Therefore we have *no* information overcount. For Smith-Waterman we have again *no* information overcount, because after phase (2) we consider only the best local alignment pattern that is comparing different characters in each position. For Spectrum we could have at most $n-k+1$ shared k-mers between s_1 and s_2. Thus, in this case, in s_2 we have at most a coverage of k times (given by k-mers) for each of the $n-2(k-1)$ central positions, and at most a coverage of $2\sum_{i=1}^{k-1} i$ times for all leading $k-1$ and all trailing $k-1$ characters. Given that a coverage without repetitions considers each shared position only once, we have a total information overcount of $(k-1)(n-2(k-1))+2\sum_{i=1}^{k-2} i = (k-1)(n-k)$, hence $O(kn)$. For Word Correlation we have the same maximum value of information overcount as for Spectrum, because in the evaluation of pairwise similarity between the k-mers of s_1 and s_2 we consider the comparison of a k-mer with another k-mer only once. Thus the output repetitions are based on the overlaps between the shared k-mers. In Mismatch we have the information overcount of Spectrum plus an additional redundancy due to the spreading of the number of occurrences of a k-mer to the other k-mers within e errors. The last part of the summation can be estimated in $k(n-k+1)\sum_{i=1}^{e}\binom{k}{i}(|\Sigma|-1)^i$, where the factor k is the number of positions covered by each k-mer, $n-k+1$ is the maximum number of shared k-mers, and the last factor is the number of k-mers within e errors

TABLE 2.3. COMPARISON OF THE INFORMATION OVERCOUNT

Algorithm	Information Overcount
Irredundant Class	*none*
Smith-Waterman	*none*
Spectrum (k)	$O(kn)$
Word Correlation (k)	$O(kn)$
Mismatch (k, e)	$O(k^{e+1}\|\Sigma\|^e n)$
Local Alignment	$O(n^3)$

Comparison of algorithms using the Information Overcount Model, where m and $n \leq m$ are the lengths of the sequences s_1 and s_2. Rows are listed from the best to the worst result.

from a fixed k-mer. Then, the resulting information overcount would be $(k-1)(n-k) + k(n-k+1)\sum_{i=1}^{e}\binom{k}{i}(|\Sigma|-1)^i = O(k^{e+1}|\Sigma|^e n)$. Finally, in Local Alignment we compute a global alignment for each pair of substrings of s_1 and s_2. Thus the information overcount will be based only on the overlaps of these substrings in s_2, resulting in $n(n+1)(n+2)/6$ repeated outputs, that is $O(n^3)$.

In Table 2.4 we present a comparison of pairwise computational complexity for the six algorithms described above, to give an idea of trade-off between information overcount (see Table 2.3), computational complexity, and effectiveness in the classification of protein sequences (see Table 2.6). These values were taken from the original articles.

2.4 Experimental Results

2.4.1 *Comparison with State-of-The-Art Methods and Statistics*

We assess the effectiveness of the Irredundant Class method on the classification of protein families into superfamilies. This problem refers to the detection of sequence homology in evolutionarily-related proteins with low-sequence similarity, and is called *remote homology detection.*

Table 2.4. Comparison of the Pairwise Complexity

Algorithm	Pairwise Complexity
Spectrum (k)	$O(kz)$
Word Correlation (k)	$O(k^2\|\Sigma\|^2 z)$
Mismatch (k, e)	$O(k^{e+1}\|\Sigma\|^e z)$
Local Alignment	$O(z^2)$
Smith-Waterman	$O(z^2)$
Irredundant Class	$O(z^2 \log z \log \|\Sigma\|)$

Comparison of algorithms based on their pairwise computational complexity, where $z = m + n$. Rows are listed from the best to the worst result. Although $|\Sigma|$ is a constant in protein analysis, its value can significantly affect the complexity being about 20, and thus it is here considered.

Tests are based on the dataset of Liao and Noble described in [74],[2] which uses the Structural Classification Of Proteins (SCOP)[3] of [80], version 1.53, as a reference. The dataset consists of 4,352 sequences of about 560,000 amino acids in total, which are grouped by SCOP into 54 families and 23 superfamilies. For each family, proteins within the family are considered positive test examples, and proteins within the superfamily but outside the family are considered positive training examples; negative examples are chosen outside the fold, and were randomly split into training and test sets in the same ratio as the positive examples. Therefore this assessment consists of 54 experiments, each corresponding to a target family and having at least 10 positive training examples taken from its respective superfamily, and 5 positive test examples taken directly from the family, with no sequence homology known a priori. These experiments there are usually very unbalanced, with a much larger number of negative examples than of positive examples, as illustrated in Table 2.5, and with lengths of sequences that ranges from 20 to about 1,000 amino acids. In short, the task consists on classifying target families of sequences into superfamilies of SCOP using an SVM.

[2]The dataset is available at `http://noble.gs.washington.edu/proj/svm-pairwise`.

[3]SCOP, a protein classification manually constructed by visual inspection and comparison of structures, is available at `http://scop.mrc-lmb.cam.ac.uk/scop`.

Table 2.5. Experiments of Liao and Noble

		Training		*Test*				*Training*		*Test*	
No.	Target family	Pos.	Neg.	Pos.	Neg.	No.	Target family	Pos.	Neg.	Pos.	Neg.
0	7.3.5.2	12	2330	9	1746	27	7.3.10.1	11	423	95	3653
1	2.56.1.2	11	2509	8	1824	28	3.32.1.11	46	3880	5	421
2	3.1.8.1	19	3002	8	1263	29	3.32.1.13	43	3627	8	674
3	3.1.8.3	17	2686	10	1579	30	7.3.6.1	33	3203	9	873
4	1.27.1.1	12	2890	6	1444	31	7.3.6.2	16	1553	26	2523
5	1.27.1.2	10	2408	8	1926	32	7.3.6.4	37	3591	5	485
6	3.42.1.1	29	3208	10	1105	33	2.38.4.1	30	3682	5	613
7	1.45.1.2	33	3650	6	663	34	2.1.1.1	90	3102	31	1068
8	1.4.1.1	26	2256	23	1994	35	2.1.1.2	99	3412	22	758
9	2.9.1.2	17	2370	14	1951	36	3.32.1.1	42	3542	9	759
10	1.4.1.2	41	3557	8	693	37	2.38.4.3	24	2946	11	1349
11	2.9.1.3	26	3625	5	696	38	2.1.1.3	113	3895	8	275
12	1.4.1.3	40	3470	9	780	39	2.1.1.4	88	3033	33	1137
13	2.44.1.2	11	307	140	3894	40	2.38.4.5	26	3191	9	1104
14	2.9.1.4	21	2928	10	1393	41	2.1.1.5	94	3240	27	930
15	3.42.1.5	26	2876	13	1437	42	7.39.1.2	20	3204	7	1121
16	3.2.1.2	37	3002	16	1297	43	2.52.1.2	12	3060	5	1275
17	3.42.1.8	34	3761	5	552	44	7.39.1.3	13	2083	14	2242
18	3.2.1.3	44	3569	9	730	45	1.36.1.2	29	3477	7	839
19	3.2.1.4	46	3732	7	567	46	3.32.1.8	40	3374	11	927
20	3.2.1.5	46	3732	7	567	47	1.36.1.5	10	1199	26	3117
21	3.2.1.6	48	3894	5	405	48	7.41.5.1	10	2241	9	2016
22	2.28.1.1	18	1246	44	3044	49	7.41.5.2	10	2241	9	2016
23	3.3.1.2	22	3280	7	1043	50	1.41.1.2	36	3692	6	615
24	3.2.1.7	48	3894	5	405	51	2.5.1.1	13	2345	11	1983
25	2.28.1.3	56	3875	6	415	52	2.5.1.3	14	2525	10	1803
26	3.3.1.5	13	1938	16	2385	53	1.41.1.5	17	1744	25	2563

Experiments presented in [74] and associated to 54 protein families of SCOP version 1.53, here ordered by progressive number. For each target family ID is detailed the number of positive and negative sequences of training and test.

We compared the Irredundant Class with the state-of-the-art methods in remote protein homology detection, employing as metric the receiver operating characteristic score (ROC score). For each experiment, given a ranking of the test sequences scores in output from the SVM, the ROC score is the normalized area under the curve that plots the number of true positive examples found as a function of the number of false positive examples detected [46]. In practice, we plot the number of true and false positives found in a two-dimensional histogram, with the false positives in abscissa and the true positives in ordinate, for each different possible classification threshold based on SVM scores (see Figure 2.3 for an example).

All methods compared are of discriminative nature, so we used a popular

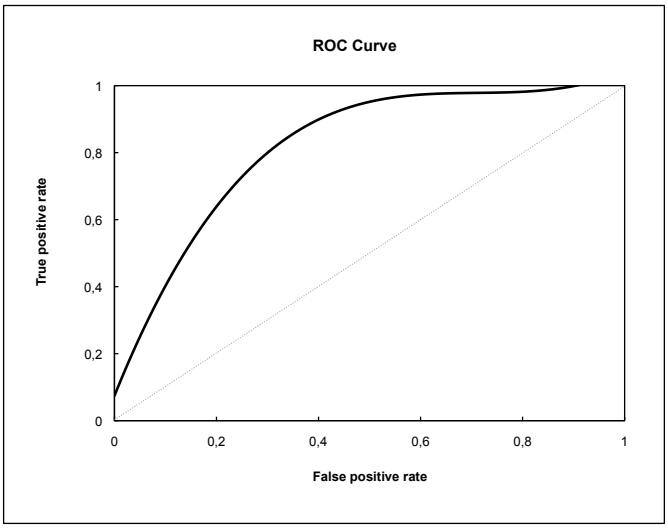


FIGURE 2.3. *Example of ROC curve used in our experiments for the family no. 16 of Table 2.5, where the dashed line represents the trend of a random projection.*

SVM software: the Gist SVM[4] described in [84], version 2.3. Experimental results of the other methods were taken from [75; 98].[5]

Table 2.6 shows the mean ROC scores, that is the average of ROC scores over all experiments, for the Irredundant Class and the other methods. These scores indicate that our method outperforms most methods in literature, and it is comparable to the state-of-the-art Local Alignment. For a more detailed view, the ROC scores distribution is illustrated for some methods in Figure 2.4. The Local Alignment in green color (triangles) seems to perform slightly better than the Irredundant Class in blue (squares), but the minimum ROC score of the Local Alignment is much smaller. In particular, the smallest ROC score of our method was obtained in experiment 15 of [74] with a value of 0.614, while all other methods got their lowest peaks in experiment 13 with very small values, for example 0.22 for the Local Alignment. To confirm this fact, Figure 2.5 reports the ROC scores distribution showing in detail the trend for all experiments, and evidencing that the Irredundant Class exhibits, in general, a more robust behavior than the other methods.

[4]Gist SVM is available at `http://www.bioinformatics.ubc.ca/gist`.

[5]Details on the state-of-the-art results are available at
`http://sunflower.kuicr.kyoto-u.ac.jp/~hiroto/project/homology.html`.

TABLE 2.6. COMPARISON OF EXPERIMENTAL RESULTS

Method	Mean ROC
Irredundant Class	0.929
Local Alignment (version "eig")	0.925
Word Correlation Matrices (6)	0.904
Pairwise	0.896
Mismatch (5,1)	0.872
Spectrum (3)	0.824
Fisher	0.773

Comparison of state-of-the-art methods based on their mean ROC score over all experiments. Rows are listed from the best to the worst.

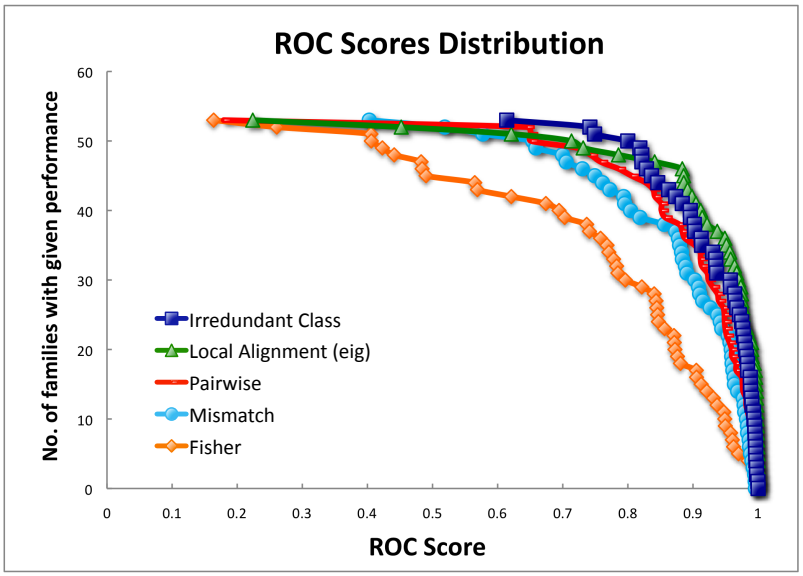


FIGURE 2.4. *ROC scores distribution for the Irredundant Class and the state-of-the-art methods.*

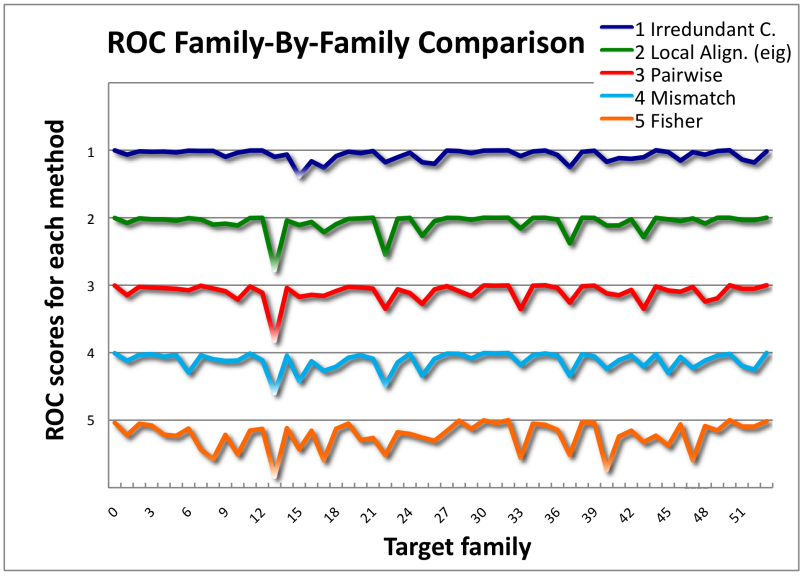
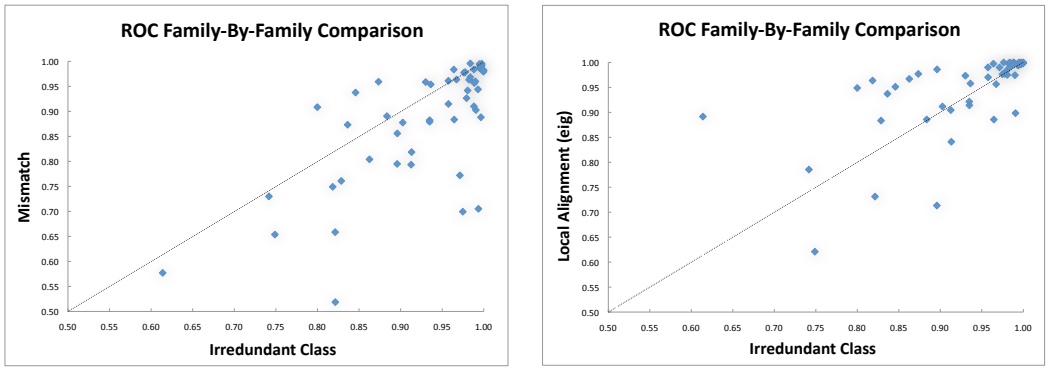


FIGURE 2.5. *ROC scores across experiments.*

Family-by-family details can be caught from Figure 2.6 for the comparison of the Irredundant Class versus, respectively, (a) Mismatch and (b) Local Alignment ("eig"). The former is one is known to achieve high performance, while the latter is one the most effective kernels in literature. Again, we can observe that Irredundant Class achieves better performance than Mismatch, and similar results when compared with Local Alignment.

Finally, Table 2.7 reports the number of irredundant common motifs versus a less restrictive notion of patterns, the maximal common motifs previously introduced in the chapter (see Definition 2.3), for 10 pairs of protein sequences randomly chosen from our experiments. Results indicate that the number of irredundant common motifs $\mathcal{I}_{s_1,s_2}$ tends to be an order of magnitude smaller than the number of maximals $\mathcal{M}_{s_1,s_2}$, except for very short sequences (see pair numbers 9 and 10 of Table 2.7).

Table 2.7 evidences that, slightly relaxing the notion of irredundancy — for example considering the maximal common motifs, denoted by $\mathcal{M}_{s_1,s_2}$ and in relation $\mathcal{I}_{s_1,s_2} \subseteq \mathcal{M}_{s_1,s_2}$,— we could have a number of patterns that grows exponentially with the length of the sequences; while the irredundants are, in every case, less than $m+n-3$, thus avoiding the comparison of the same pair of characters of s_1 and s_2 a multiple number of times (see Section 2.3 for a



(a) (b)

FIGURE 2.6. *(a) Family-by-family ROC scores comparison of the Irredundant Class versus Mismatch. (b) Family-by-family ROC scores comparison of the Irredundant Class versus Local Alignment version "eig."*

theoretical review). Furthermore, maximal common motifs can be prohibitive to extract for some long sequences. Whereas we have already proved that the number of irredundant common motifs is at most linear in the size of sequences.

2.4.2 *Analysis of the Irredundant Class Information Content*

Although the classification of protein sequences, through an SVM, does not provide an alignment per se, one can use the footprint of irredundant common motifs to detect candidate functional sites in protein sequences. We are not interested in aligning a set of sequences, but we want to analyze the distribution of the most discriminative irredundant common motifs.

We recall that the result of the SVM learning process, for a target protein family, is a set of weights $\alpha = (\alpha_1, \ldots, \alpha_N)$ associated to the N training sequences of its superfamily. We want to study the distribution of irredundant common motifs in the test sequences, using for each pattern p a weight that is proportional to its score $\frac{F_p}{E[F_p]}$ and to the weight α_i of the corresponding training sequence that generated p, with $1 \leq i \leq N$. Consider a test sequence s_{test} and the set of training sequences $s_1, \ldots, s_N$; each pair (s_{test}, s_i) generates a set of irredundant common motifs $\mathcal{I}_{test,i}$. For each pattern p in $\mathcal{I}_{test,i}$ we compute its score as the product $\ln(\frac{F_p}{E[F_p]}) \times \alpha_i$ and we associate this score

TABLE 2.7. MAIN COUNTS FOR THE IRREDUNDANT COMMON MOTIFS

No.	s_1	s_2	m	n	$m+n$	$\lvert\mathcal{M}_{s_1,s_2}\rvert$	$\lvert\mathcal{I}_{s_1,s_2}\rvert$	% $\mathcal{I}_{s_1,s_2}$
1.	1alo_4	1bjt__	597	760	1,357	16,697	1,256	7.5
2.	1qaxa2	1cxp.1c	316	466	782	8,397	682	8.1
3.	1gai__	1nmta2	472	227	699	7,037	612	8.7
4.	1cvua1	1lgr_2	511	368	879	9,014	787	8.7
5.	1gpea1	1yrga_	392	343	735	6,853	653	9.5
6.	1qqja_	3pccm_	415	236	651	5,090	566	11.1
7.	1bxka_	1ofga1	352	220	572	3,549	489	13.8
8.	1ebfa1	2naca1	169	188	357	1,126	277	24.6
9.	1a03a_	1mho__	90	88	178	257	108	42.0
10.	1gpt__	1ayj__	47	50	97	64	45	70.3

Number of irredundant ($\mathcal{I}_{s_1,s_2}$) versus maximal ($\mathcal{M}_{s_1,s_2}$) common motifs over 10 pairs of protein sequences taken from experiments. Rows are listed according to the percentage of irredundants over the number of maximals, where $\mathcal{I}_{s_1,s_2} \subseteq \mathcal{M}_{s_1,s_2}$.

to the positions of s_{test} covered by solid characters of p. We repeat the same process for all patterns in $\mathcal{I}_{test,i}$, for each $1 \leq i \leq N$; and for each location we sum the contributions of all patterns that cover that location. We obtain a histogram of the footprint of the irredundant common motifs for the test sequence s_{test}. The interpretation of this histogram is that conserved regions should correspond to some picks, thus to candidate functional sites detected by the Irredundant Class.

We picked three families at random from the dataset used in our experiments. For every protein family we use as target functional sites the PROSITE [53] manually found consensus patterns. To better highlight the distribution of footprints, we build, for each family, a multiple alignment of the test sequences and plot all histograms over this alignment. Figure 2.7 shows the resulting histogram for the protein family S100. A red bar shows the location of the functional pattern reported by PROSITE, also shown in the picture. For this family we can see that a clear signal is present, and that some picks correspond quite well with conserved amino acids in the func-

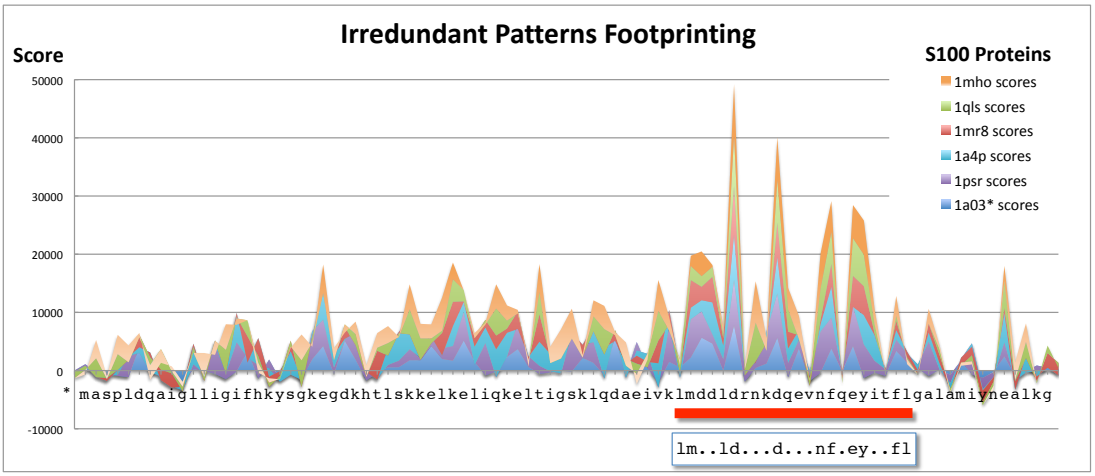
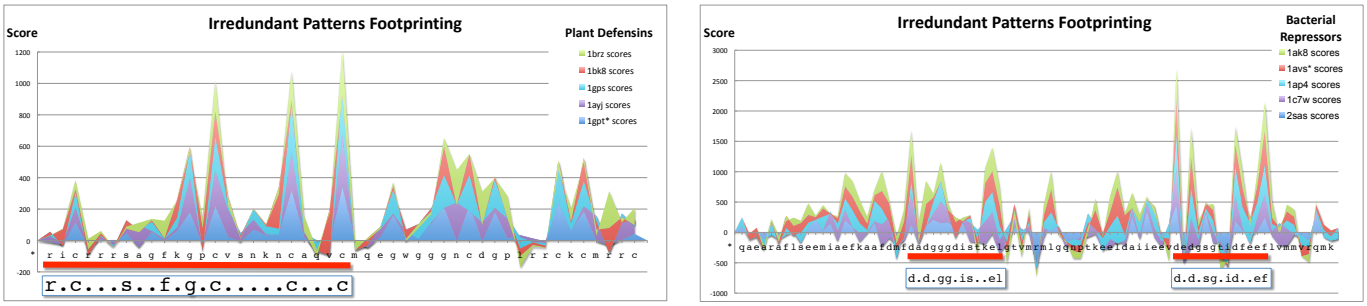


FIGURE 2.7. *Histogram of the irredundant patterns footprint, which accounts for multiple irredundant common motifs, for S100 proteins (family no. 50 of Table 2.5).*

tional site. Similar considerations apply also for the families in Figure 2.4.2. In Figure 2.4.2(a) we observe picks mostly in correspondence of Cysteines, whereas in Figure 2.4.2(b) the pattern reported by PROSITE results in two functional sites that share comparable high scores.

Note that these results are obtained comparing sequences from a protein family and its superfamily, thus the chance to find the actual signal is weak as opposed to standard alignment methods, that consider only the protein family. Nevertheless, figures 2.7 and 2.4.2 indicate that the profile of the family functional site is retained by the irredundant common motifs, and may be computed as a post process of our analysis, as we will see in the next chapters.

This analysis does not yield to an alignment of sequences, but it is a way to interpret the distribution of the Irredundant Class. In summary the most discriminative patterns contain information about the functional site of a protein family, but this information is not explicitly available by simple inspection.



(a) (b)

FIGURE 2.8. *Histogram of the irredundant patterns footprint for: (a) plant defensins (family no. 32 of Table 2.5) and (b) bacterial repressors (family no. 53).*

2.5 Discussion and Future Work

In this chapter we have studied how the notion of irredundant common motifs can solve an issue that is rising in the field of string kernels, the remote homology detection. Almost all string kernels are based on patterns that are not independent, and therefore the associated scores are obtained using a set of redundant features. We have specifically addressed this issue considering a particular class of patterns, the irredundant common motifs with don't cares. The method, called Irredundant Class, is based on the statistics of these patterns and the ability to discriminate given by the support vector machines. Results on benchmark data show that the Irredundant Class outperforms most of the string kernels previously proposed, and achieves results as good as the current state-of-the-art method Local Alignment.

Despite its information properties, the Irredundant Class has a computational complexity that is much more than linear in the size of sequences. This issue can be partially solved by a suitable parallelization made available by the discovery procedure of these patterns (Algorithm 2.1).

In addiction, our method processes the same characters of a sequence a multiple number of times, due to pattern overlaps. Therefore, in the next chapters we plan to study a new notion of meaningful patterns able to manage these issues and to better fit the homology detection problem in protein sequences.

Chapter 3

Comparing, Ranking, and Filtering Degenerate Motifs for Identifying Subtle Biological Signals

In biology the notion of degenerate motif plays a central role for describing various phenomena. For example, protein functional motifs, like the ones contained in the PROSITE database [53], e.g., `[FY]DPC[LIM][ASG]C[ASG]`, are in general represented with character classes, or degenerate motifs, denoting conserved sites in protein families. These complex motifs are collected using semi-automatic procedures, nevertheless they are still manually verified.

The discovery of degenerate motifs in protein and genome sequences is becoming increasingly important [55; 86]. Such motifs usually correspond to residues conserved during evolution due to some significant structural or functional role. Moreover, the increasing availability of biological sequences, recently achieved with high-throughput sequencing technologies and a multitude of new protein discoveries, has increased the number and complexity of motifs required by scientists in order to perform a complete analysis of some biological issues.

In order to fill this gap, researchers have developed several approaches over the years. Typically these approaches follow in popular frameworks of degenerate motif discovery [8; 42; 55; 107]. Although such methods have been exhaustively consolidated and successfully tested on small genomes and

set of proteins, their outcome often far exceeds the size of the original input, making it impractical to be managed and then interpreted for further analysis requiring a manual inspection.

In this chapter we address the problem of degenerate motif comparison and filtering, in order to simplify the output of some state-of-the-art motif discovery tool for the identification of subtle biological signals. This could also serve as a possible integration of multiple discovery tools.

We first give a characterization of degenerate motifs, in particular of motifs with character classes without gaps, following with the notion of *motif priority* for comparing and ranking different motifs together. Then, we introduce the concept of *underlying motifs* for filtering any set of motifs with character classes into a new set that is linear in size with respect to a reference sequence s, that could be the concatenation of multiple sequences. Each underlying motif will have the highest priority in some regions of s, and thus represents these regions in our framework. We finally discuss some experimental results on the identification of subtle signals in protein families by means of the underlying motifs, in conjunction with the concept of motif priority. Results show that our approach drastically reduces the number of motifs in output from a state-of-the-art tool in protein analysis, retaining and ranking the actual functional motifs in the top 5 out of the thousands of motifs found.

3.1 Background

3.1.1 Motif Representation

In general, a motif is represented by an ordered sequence of symbols and a set of locations where it appears. A motif, along with its list of occurrences, is usually extracted from one or more sequences over an alphabet Σ.

To model mutations and other evolutionary events, a number of formalisms have been introduced. For example, position-specific scoring matrices (PSSMs, [17]) are widely used to model transcription factor binding sites, where each score represents the probability, or number of times, that a given symbol appears at a certain position. As an instance Figure 3.1 shows

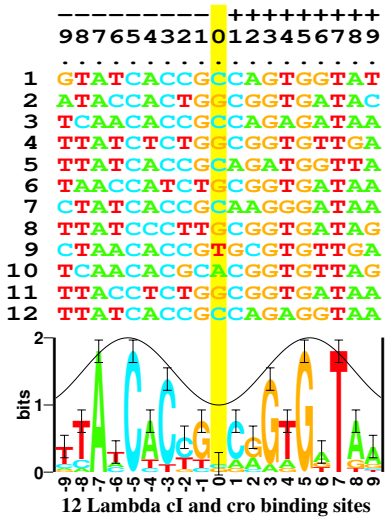


FIGURE 3.1. *Example of sequence alignment logo taken from [100].*

a multiple alignment of sequences generating a PSSM.

Another popular model are character classes, like the IUPAC[1] code [30]. IUPAC specifies alphabets for proteins, DNA, and also for RNA. Moreover, for protein sequences a number of other clustering approaches have been proposed [34; 50; 53; 68]. For example, the PROSITE motif for the Nickel-dependent hydrogenases large subunit signature is `RG[LIVMF]E`·····`[QESMP][RK]·C[GR][LIVM]C`.

In this study we are interested in motifs with character classes, also called *degenerate motifs* (with no gaps). Often the classification of symbols does not completely partition an alphabet, and also in practical cases motifs show classes with non-empty intersections. For example, in the above motif there are intersections between classes, like that of `[RK]` with `[GR]`, and also subsets, like `[LIVM]` with respect to `[LIVMF]`. If the motifs in input contain only partitions of the original alphabet Σ, we can map the input sequences into the new alphabet of symbols represented by the partitions, thus considerably reducing the number of candidate motifs to be analyzed (see for example [112]). In all other cases this will result in the combinatorial explosion of candidate

[1]IUPAC, International Union of Pure and Applied Chemistry, projects and publications are available at `http://www.iupac.org`.

motifs, that consequently affects the efficiency of matching and discovering new motifs. In this chapter we will set the basis to solve this fundamental problem.

3.1.2 Ranking and Clustering Degenerate Motifs

To cope with the combinatorial explosion of motifs, motif discovery tools must first shrink the search space. In this regard, a number of different techniques have been proposed over the past two decades [8; 42; 55; 107], on the basis of which other works have been made [22; 79; 88; 117; 118; 122]. Despite that, the number of motifs in output still remains in most cases intractable.

Moreover, all motif discovery tools must ultimately rank the output, according to some measure of importance. For example, even if very sophisticated probabilistic scoring mechanisms are in place in a state-of-the-art tool for protein analysis such as Varun [8], there are cases where the direct interpretation of results is far from trivial. Let us consider, for instance, the worst case of Varun presented in Figure 3.2. The functionally relevant motif, shown in bold, appears at position 42, whereas almost all top motifs are somehow very similar. In order to filter out this output and enhance its readability, there are two main issues. On the one hand, most motifs are very similar in sequence and therefore they must be clustered together. On the other hand, if we are interested in a specific region of some considered sequences, one would like to select the most important motif that appears in that region according to some rule. These two issues are in practice tightly related and need to be addressed as one.

Recently, a number of ensemble methods addressing some of these issues have been proposed [24; 96; 121; 126]. The basic idea is that one can integrate the outcome of several motif discovery algorithms based on sophisticated heuristics, in order to improve the ability of finding subtle biological signals in specific contexts.

The ensemble methods actually rank all predicted motifs according to some scoring functions, and then report the top motifs. WebMotifs [96], ARCS-Motif [126], MotifMiner [24], and MotifVoter [121] assume that some

Rank	z-score	Motif
1	2,84E+09	Y...L...C..[FYW]A..[STAH]R..P..FNE[STAH]K.I.F[STAH]M
2	8,28E+07	V-(1,3,4)G...S..[STAH]....N...L....Q-(4)[STAH]....L.[DN]...[FYW]..F....P....Q..A...I
3	5,55E+07	L-(2,3)F...Q....[STAH][STAH]...L.[DN]...[FYW]..F.R..P.D..Q..A...I
4	4,27E+07	L-(2,3)F...Q.[STAH]..[STAH][STAH]....S....[FYW]..F.R..P.D..Q..A...I
5	4,23E+07	L....I...[STAH]..[STAH]....LS[DN]...[FYW]..F.R..P.D..Q..A...I
6	3,99E+07	LF-(3)Q....[STAH][STAH]....S[DN]...[FYW]..F.R..P.D..Q..A...I
7	3,38E+07	LF-(3)Q....[STAH][STAH]...L.[DN]...[FYW]..F.R..P.D..Q..A...I
8	3,38E+07	LF...Q....[STAH]-(4)L.[DN]...[FYW]..F.R..P.D..Q[STAH].A...I
9	3,29E+07	I-(1)Q.[STAH]..[STAH]....LS[DN]...[FYW]..F.R..P.D..Q..A...I
10	3,29E+07	I.Q-(4)[STAH]....LS[DN]...[FYW]..F.R..P.D..Q[STAH].A...I
11	3,29E+07	I.Q.[STAH]..[STAH]-(4)LS[DN]...[FYW]..F.R..P.D..Q..A...I
12	3,10E+07	L....Q-(1,4)[STAH]..[STAH]....LS[DN]...[FYW]..F.R..P.D..Q..A...I
13	2,77E+07	L[FYW]-(3)Q.[STAH]..[STAH]....LS....[FYW]..F.R..P.D..Q..A...I
14	2,58E+07	L-(4)Q.[STAH]..[STAH]....LS[DN]...[FYW]..F.R..P.D..Q..A...I
15	2,30E+07	S.[STAH]S-(2,4)LS[DN]...[FYW]..F.R..P.D..Q[STAH].A...I
16	2,15E+07	L-(1,3,4)C..[FYW]A..[STAH]R..P..F.E.K.I.F.M
17	1,40E+07	F-(1)I.Q...[STAH][STAH]-(4)L[STAH]....[FYW]..F.R..P.D..Q..A...I
18	1,37E+07	L-(2,4)I...[STAH].[STAH].[STAH]-(3)LS....[FYW]..F.R..P.D..Q..A...I
19	1,02E+07	L..I-(1)Q....[STAH][STAH]....S....[FYW]..F.R..P.D..Q..A...I
20	8,65E+06	I-(1)Q....[STAH][STAH]...L.[DN]...[FYW]..F.R..P.D..Q..A...I
21	8,19E+06	S[STAH]-(1,2,3,4)LS[DN]...[FYW]..F.R..P.D..Q[STAH].A...I
22	7,98E+06	Q-(3)[STAH][STAH]....LS[DN]...[FYW]..F.R..P.D..Q..A...I
23	6,82E+06	F-(3)Q....[STAH][STAH]...L[STAH]....[FYW]..F.R..P.D..Q..A...I
24	5,66E+06	A[STAH][STAH]-(2,3)LS[DN]...[FYW]..F.R..P.D..Q..A...I
25	5,57E+06	F.I-(3)[STAH]..[STAH]....L[STAH]....[FYW]..F.R..P.D..Q..A...I
26	5,18E+06	L.L-(4)Q....[STAH]....L-(1)[DN]...[FYW]..F.R..P.D..Q..A...I
27	3,61E+06	L.L-(2)I...[STAH]...[STAH]....[STAH]....[FYW]..F.R..P.D..Q..A...I
28	3,48E+06	[STAH].[STAH]-(1,2,3)LS[DN]...[FYW]..F.R..P.D..Q..A...I
29	3,17E+06	[STAH]...[STAH]...LS[DN]...[FYW]..F.R..P.D..Q..A...I
30	2,47E+06	L....Q-(4)[STAH][STAH]....S....[FYW]..F.R..P.D..Q..A...I
31	2,43E+06	V-(1,3)N.L....I-(3)[STAH]...[STAH]....[STAH]....[FYW]..F....P.D..Q..A...I
32	2,22E+06	[STAH][STAH][STAH]-(1,2,3)LS....[FYW]..F.R..P.D..Q..A...I
33	2,06E+06	[STAH].[STAH][STAH]....LS....[FYW]..F.R..P.D..Q..A...I
34	2,03E+06	Y...L...C...A...R..P..F.E.K.I-(1,4)[FYW][STAH]
35	1,99E+06	I.Q...[STAH]-(1)[STAH]...L.[DN]...[FYW]..F....P.D..Q..A...I
36	1,99E+06	I.Q-(1)[STAH]...[STAH]...L.[DN]...[FYW]..F....P.D..Q..A...I
38	1,97E+06	F.I...[STAH]-(3)[STAH]...L.[DN]...[FYW]..F....P.D..Q..A...I
40	1,97E+06	F.I-(3)[STAH]..[STAH]....L.[DN]...[FYW]..F....P.D..Q..A...I
41	1,91E+06	[STAH]..[STAH].K-(1,4)P..FNE[STAH]K.I.F[STAH]M
42	**1,72E+06**	**CC[FYW].C..C....[FYW]-(2,4)[DN]..[STAH]C..C**
43	1,57E+06	[STAH]-(1,3,4)[FYW]A..[STAH]R..P..F.E.K.I.F.M
44	1,49E+06	A-(1,3)[STAH]...L[STAH][DN]...[FYW]..F.R..P.D..Q..A...I
45	1,36E+06	Q...[STAH].[STAH]-(3)L[STAH]....[FYW]..F.R..P.D..Q..A...I
46	1,32E+06	I-(3)[STAH]..[STAH][STAH]....S....[FYW]..F.R..P.D..Q..A...I
47	1,31E+06	[STAH][STAH]-(1,2,3,4)L.[DN]...[FYW]..F.R..P.D..Q..A...I
48	1,24E+06	[STAH]..[STAH][STAH]-(1,3)LS....[FYW]..F.R..P.D..Q..A...I
49	1,19E+06	[FYW]-(1,3,4)[STAH]...P..FNE[STAH]K.I.F[STAH]M
50	1,12E+06	I...[STAH]-(3)[STAH]...L[STAH]....[FYW]..F.R..P.D..Q..A...I

FIGURE 3.2. *Output of Varun for the G-protein coupled receptors family 3 (id PS00980), consisting in 25 sequences of about 25,000 amino acids each. One of the functionally relevant motifs is shown in bold among the first 50 extensible motifs —i.e., with a variable number of don't cares per site— ordered by Z-Score. Illustration taken from [8].*

of the motifs given by the consensus of several state-of-the-art finders are likely to be the actual functional motifs. They ultimately cluster all motifs and report only those from the best clusters. However, if none of the motifs from the individual finders can accurately capture the binding sites, the performance of the ensemble methods will suffer. Although these methods indeed help to improve the performance of motif finding, the improvement is usually not significant. For example, in Tompa's benchmark [111] and Escherichia coli datasets, the average sensitivity is only improved by 62 %, but the average precision is reduced by 15 %.

This relatively new topic deserves more attentions, along with its mathematical and combinatorial foundations. In this chapter we discuss the basic problems behind motifs comparison and ranking. We will show how simple lexicographic properties can help to select a small subset of motifs, therefore filtering the output of any finder to a more tractable size.

3.1.3 Roadmap of the Work

The main problem we want to address here is the following:

Given a reference sequence s, that could be the concatenation of multiple sequences, and a set of motifs with character classes $\mathcal{M}$*, that is the outcome of one or more motif finders, we aim to reduce the set $\mathcal{M}$ to a small number of ordered representative motifs, say $\mathcal{U}$, such that the size of $\mathcal{U}$ is linear with respect to the reference sequence s.*

This reduction will allow us to manage the new set easier, and will enhance its readability and interpretation. The general idea behind this work is to identify interesting lexicographic properties of degenerate motifs that are useful for filtering any set of these motifs and thus detecting subtle biological signals. In particular, we will prove that some simple properties are capable of ranking any set $\mathcal{M}$ of motifs with character classes, and also to select a representative subset of $\mathcal{M}$, which we call the *underlying representative set* $\mathcal{U}$. Inherently, with the notion of underlying motifs we want to discard those motifs that rank lower than others if located in the same occurrences on the reference sequence s.

Thus, we first characterize motifs with character classes defining the no-

tion of minimality, that makes the motifs more specific with respect to the reference sequence s. Then, we define the notion of *motif priority*, that permits to compare and rank different motifs together. The priority is defined using properties like the motif length, composition (accounting also for character classes), and the lexicographic order of occurrences. In particular, these properties have been specifically designed for protein sequences by analyzing the results of Chapter 2 (Subsection 2.4.2) and of [8]. In the last part of the chapter, we will show that the motif priority rule, together with the notion of underlying motifs, can filter a set of degenerate motifs $\mathcal{M}$ into a new set of linear size with respect to s. Moreover, our rule empirically proved to be useful for filtering biological motifs, in order to ultimately identify conserved functional sites.

We believe that such an approach can drastically reduce the number of candidate motifs to be analyzed in genomics and proteomics, and can enhance their readability and interpretation. We present a series of results on protein families that support the validity of the theoretical findings. These preliminary experiments show very good performance for our approach as a filter to reduce the number of motifs while keeping the most important ones.

3.2 Motifs with Character Classes: a Characterization

As introduced in Chapter 2, a *string* is a sequence of zero or more symbols from an alphabet $\mathcal{A}$, and the set of all strings over $\mathcal{A}$ is denoted by $\mathcal{A}^*$. The length of a string x is defined as the number of its symbols and denoted by $|x|$; in this case, a character class accounts for one symbol. The i-th symbol of x will be denoted by x_i or by $x[i]$, where $1 \leq i \leq |x|$.

Let Σ be a finite alphabet. For example, in genomics Σ corresponds to the set of nucleotides {A,C,G,T} or {A,C,G,U}, while in proteomics Σ corresponds to the set of the 20 amino acids. Then, a symbol σ from Σ is called a *solid character*. Throughout the chapter a string s composed of solid characters from Σ will be used as reference sequence. Conversely, motifs will be represented by a more expressive alphabet, as we explain in the following definitions.

Definition 3.1. *(Character class) Let R_Σ be a set of equivalence relations over Σ. A* character class *is a subset of Σ consisting of two or more equivalent solid characters with respect to some relation in R_Σ.*

For instance, the hydrophobicity property can partitioning all the 20 amino acids into disjoint classes. In this regard, a number of different hydrophobicity scales have been developed over the years [20]. A hydrophobicity scale thus generates an equivalence relation $R \in R_\Sigma$ between solid characters, such that each class of the resulting quotient set Σ/R can be defined as a character class. It is possible to define several character classes over Σ, according to the equivalence relations in R_Σ. In particular, we consider the set of maximal character classes on R_Σ, i.e., those classes that are not proper subsets of other classes. This argument will be further discussed in Subsection 3.2.1.

Given τ maximal character classes $\{C_i\}_1^\tau$ on Σ, $\bigcup_{i=1}^{\tau} 2^{C_i}$ represents all non-empty subsets of the classes.

Definition 3.2. *(Pattern) A* pattern with character classes*, or simply a* pattern*, is a string defined on $\Sigma \cup (\bigcup_{i=1}^{\tau} 2^{C_i})$.*

In the literature, this type of degenerate patterns has been extensively studied, and is sometimes also called an *indeterminate string* [2; 41; 81; 90; 91; 110; 123]. Given a pattern $p = p_1 p_2 \dots p_k$ of length k, each symbol p_j, with $1 \leq j \leq k$, is either a solid character or a subset of a character class C. In the latter case, we write p_j by means of square brackets, as the PROSITE convention [53]: $p_j = [\sigma_1, \sigma_2, \dots, \sigma_{r_j}]$.

Definition 3.3. *(Pattern occurrence) A pattern $p = p_1 p_2 \dots p_k$ of length k is said to* occur *at a location l of a string s, with $1 \leq l \leq n$, if $s_{l+j-1} \in p_j$ for each $1 \leq j \leq k$.*

Table 3.1 shows an example of pattern and pattern occurrences, where $s =$ `AABEADBACE`, $\Sigma = \{$`A,B,C,D,E`$\}$, and the character classes are $C_1 = \{$`A,C,D`$\}$ and $C_2 = \{$`A,B,E`$\}$. In this case, $p =$ `A[A,C,D][B,E]` is a pattern of length $k = 3$ that occurs at locations $1, 5, 8$ of s.

Hereafter we will assume that we are given a string s on Σ of length n, a set of character classes $\{C_i\}_1^\tau$, with $r = \max_{1 \leq i \leq \tau}\{|C_i|\}$, and a positive

TABLE 3.1. EXAMPLE OF PATTERN WITH CHARACTER CLASSES

j	1	2	3
$p[j]$	A	[A,C,D]	[B,E]

j	1	2	3	4	5	6	7	8	9	10
$s[j]$	A	A	B	E	A	D	B	A	C	E
p	■	■	■		■	■	■	■	■	■

Pattern with character classes p, together with its occurrences $1, 5, 8$ in the string s.

integer q, $2 \leq q \leq |s|$, called *quorum*. A *motif* m is then defined as a pair $(p, \mathcal{L}_m)$, where p is a pattern of length k that occurs at the locations given by $\mathcal{L}_m$, and $\mathcal{L}_m$ is a set of at least q locations of s. For instance, as of Table 3.1, $m = (p, \mathcal{L}_m)$ is a motif with pattern $p =$ `A[A,C,D][B,E]` of length $k = 3$ and location list $\mathcal{L}_m = \{1, 5, 8\}$ of cardinality $\nu = 3$. More formally:

Definition 3.4. *(Motif, Location list) We say that $m = (p, \mathcal{L}_m)$ is a motif with pattern $p = p_1 p_2 \dots p_k$ and location list $\mathcal{L}_m = (l_1, l_2, \dots, l_\nu)$ if and only if all of the following hold:*

(i) the length of m, denoted by $|m|$, is $|p| = k \geq 2$ symbols,

(ii) for each location $l \in \mathcal{L}_m$, p occurs at l in s, and $|\mathcal{L}_m| = \nu \geq q$, and

(iii) there does not exist a location $l' \notin \mathcal{L}_m$ such that p occurs at l' in s ($\mathcal{L}_m$ is complete).

We suppose now that an oracle has given us a set $\mathcal{M}$ of interesting motifs lying on the string s. Intuitively this set may be unusually large, with the number of elements that may reach up to $O(\sum_{k=2}^{n-q+1}(\min\{\tau 2^r + |\Sigma|, 2^{|\Sigma|}\})^k) = O((\min\{\tau 2^r, 2^{|\Sigma|}\})^n) = O(2^{|\Sigma|n})$ motifs. Thus, we want to characterize the whole set of motifs $\mathcal{M}$ with *representative motifs* that have to be $O(n)$ in

number and dimensions, in order to filter out the information shared by some motifs, and enhance their readability and interpretation.

Next sections lay the foundations for finding these particular representative motifs, which we call *underlying motifs*. In loose terms, the underlying motifs are the most important motifs in $\mathcal{M}$ for each region of the string s, according to the rules we will define in the following subsections.

3.2.1 On Transitive Properties of Character Classes

Previously, we have introduced the concept of character class consisting of two or more solid characters, obtained through some equivalence relation on Σ. Every subset of character class is seen as an independent element wherewith building motifs. Then, since for patterns there is no distinction between a character class and its subsets, we consider maximal character classes as input for our problem.

Typical is the case where the considered relations R_Σ of equivalence between solid characters are not mutually transitive, since they might partition Σ in different ways [22; 117; 122]. For example, if σ_1, σ_2 are two solid characters belonging to a character class C_1, $\sigma_1, \sigma_3 \in C_2$, and $\sigma_2, \sigma_3 \in C_3$, then it could be the case that $\sigma_1, \sigma_2, \sigma_3$ do not belong to a common class C_4.

Consider, for simplicity, that R_Σ is mutually transitive on Σ. Then, the set of character classes partitions Σ, and we can map each solid character on Σ into either its maximal character class, or itself. Call $R_\Sigma(\cdot)$ this function. Accordingly, we can map $s = s_1 s_2 \dots s_n$ into $R_\Sigma(s)$, given by $R_\Sigma(s) = R_\Sigma(s_1) R_\Sigma(s_2) \dots R_\Sigma(s_n)$. In this way every pattern becomes a pattern on the new restricted alphabet given by $R_\Sigma(\Sigma)$, that is $\Sigma \cup \{C_i\}_1^\tau$. Moreover, the discovery of this type of patterns, called in the literature *exact patterns* or *subwords*, has been consolidated over the years [37; 48; 59; 71; 112]. In short, if R_Σ is mutually transitive we can find a set of classes that is a partition of Σ. We can thus map the reference sequence on this new alphabet and use standard pattern techniques. As a consequence, every motif can be uniquely identified by its length and location list, and we can conclude that, for this particular case with "simple motifs," there is no need of the notion of *minimality* that will be introduced in the next subsection.

More attention, however, deserves the general case where the transitivity does not hold. As stressed above, the main issue is that the character classes might not be a partition of Σ. Because of this, motifs are not uniquely identified by their length and location list. In other words, there may exist two motifs m and m' lying on s that have equal length and location list, but different patterns p and p', because of different character classes that compose the corresponding symbols. Since classical motif discovery algorithms find motifs over the restricted alphabet $\Sigma \cup \{C_i\}_1^\tau$, because of its practicality and ease of computation, this may lead to a large number of motifs with same location list. Thus, in the next subsection we study the notion of *minimality* specifically designed to address this issue.

3.2.2 Minimal Motifs and Motif Priority

Definition 3.5. *(Minimal representation) Given a motif m of length k, the* minimal representation *of its pattern p is a pattern $\mu(p)$ of length k with symbols $\mu(p)_j = \bigcup_{l \in \mathcal{L}_m} s_{l+j-1}$, for $1 \leq j \leq k$.*

Fact 3.1. *The minimal representation of a pattern is unique.*

Remark 3.1. *Since $\mu(p)$ is more specific than p, that is, $\mu(p)$ cannot have more occurrences than p in s, then the list of occurrences of $\mu(p)$ must be the same as for p.*

Let $\mu(m) = (\mu(p), \mathcal{L}_m)$ be the minimal representation of a motif m with pattern p, then Remark 3.1 suggests that $\mu(m)$ is consistent with the definition of a motif, i.e., the location list $\mathcal{L}_{\mu(m)}$ is complete:

Definition 3.6. *(Minimal motif) The minimal representation of m, given by $\mu(m) = (\mu(p), \mathcal{L}_m)$, is called a* minimal motif.

Computing the minimal representation of a motif is useful when a motif is defined on $\Sigma \cup \{C_i\}_1^\tau$, rather than on actual character associations. To have a more concrete idea about this concept, Table 3.2 shows an example of minimal representation $\mu(m)$ of a motif $m = (\texttt{[A,C,D][A,C,D][A,B,E]}, \{1, 5, 8\})$, with pattern $\mu(p)$ and p respectively, where the reference string is $s =$ `AABEADBACE` and the character classes are $C_1 = \{\texttt{A,C,D}\}$ and $C_2 = \{\texttt{A,B,E}\}$.

Table 3.2. Example of Minimal Representation for a Pattern

j	1	2	3
$p[j]$	`[A,C,D]`	`[A,C,D]`	`[A,B,E]`
$\mu(p)[j]$	`A`	`[A,C,D]`	`[B,E]`

Minimal representation $\mu(p)$ of the pattern p, where the reference string is $s =$ `AABEADBACE`.

In this case we say that $m' = \mu(m) = (\texttt{A[A,C,D][B,E]}, \{1, 5, 8\})$ is a minimal motif.

Let $\mathcal{M}$ be a set of motifs with character classes lying on the string s. From Fact 3.1, each motif $m \in \mathcal{M}$ has a unique minimal representation $\mu(m)$. Thus, one can easily check that the equivalence given by $\mu(m) = \mu(m')$ is an equivalence relation. The representatives of the induced partitions are the respective unique minimal motifs.

Fact 3.2. *Let us map all the motifs in $\mathcal{M}$ into the set of their minimal representations $\mu(\mathcal{M})$, where each motif $m \in \mu(\mathcal{M})$ is the minimal representation $m = \mu(m')$ of some motif $m' \in \mathcal{M}$. Then, the set of motifs $\mathcal{M}$ is partitioned into equivalence classes by the binary relation of equality between minimal representations.*

We call $\mu(\mathcal{M})$ the *minimal representative set* of $\mathcal{M}$. Since mapping $\mathcal{M}$ into $\mu(\mathcal{M})$ could mean a drastic reduction in the number of motifs, this is in practice a first step for filtering the original motifs in $\mathcal{M}$.

Remark 3.2. *Two motifs in $\mathcal{M}$, or in $\mu(\mathcal{M})$, may have the same location list; however, two minimal motifs of equal length must have different location lists.*

As a consequence of Remark 3.2, every minimal motifs in $\mu(\mathcal{M})$ can be uniquely identified by its length and location list within the minimal representative set. We can now define an important property of motifs with character classes, the *composition*, that is the sum of the number of characters within the symbols of a motif.

Definition 3.7. *(Composition of a pattern, Composition of a motif) The* composition *of a pattern p of length k is defined as $c(p) = \sum_{j=1}^{k} |p_j|$. The composition of a motif $m = (p, \mathcal{L}_m)$, denoted by $c(m)$, is defined as the composition of its pattern $c(p)$.*

For instance, given two patterns $p = \texttt{A[A,C,D][B,E]}$ and $p' = \texttt{[A,D]B[A,B]}$, their composition is $c(p) = 6$ and $c(p') = 5$, and we say that the latter is somehow more specific than p. Moreover, the composition of the motif $m = (\texttt{A[A,C,D][B,E]}, \{1, 5, 8\})$ is equal to $c(p)$, that is 6.

A fundamental point is the following definition of *priority* between motifs, a means for comparing different motifs. Here we give priority to motif length, then to motif composition, and finally to the lexicographic order of occurrences —i.e., we give priority to motifs that appear first in s.

Definition 3.8. *(Motif priority) A motif m of length k has* priority *over a motif m' of length k', denoted $m \to m'$, if and only if either $m = m'$ or*

(1) $k > k'$, or

(2) $k = k'$ and $c(m) < c(m')$, or

(3) $k = k'$, $c(m) = c(m')$, and $\min\{l \mid l \in \mathcal{L}_m \backslash \mathcal{L}_{m'}\} < \min\{l \mid l \in \mathcal{L}_{m'} \backslash \mathcal{L}_m\}$ when both minima exist.

For instance, consider $s = \texttt{AABEADBACE}$, $m = (\texttt{A[A,C,D][B,E]}, \{1, 5, 8\})$, and $m' = (\texttt{[A,D]B[A,B,E]}, \{2, 6\})$. Then, m has priority over m', written $m \to m'$, because of equal length and composition, but with different order of appearance in s: $\min\{l \mid l \in \mathcal{L}_m \setminus \mathcal{L}_{m'}\} = 1 < \min\{l \mid l \in \mathcal{L}_{m'} \setminus \mathcal{L}_m\} = 2$. If two distinct motifs m and m' have equal length and composition, but either $\mathcal{L}_m \subseteq \mathcal{L}_{m'}$ (that includes the equality case) or $\mathcal{L}_{m'} \subset \mathcal{L}_m$, then we consider m and m' *incomparable*, otherwise they are comparable. Nevertheless, we will solve this issue along with Theorem 3.2.

Given our binary relation, called *motif priority*, we want to prove that some intuitive properties hold:

Definition 3.9. *(Acyclicity) A binary relation R over a set $\mathcal{M}$ is defined to be* acyclic *if there does not exist a chain of distinct elements of $\mathcal{M}$,*

$m_1, m_2, m_3, \ldots, m_t$, such that m_1 R m_2, m_2 R $m_3, \ldots$, m_{t-1} R m_t, and also m_t R m_1 (cycle).

Definition 3.10. *(Sub-ordered set, Totally ordered set) Let R be a binary relation over a set $\mathcal{M}$. Then, $\mathcal{M}$ is said to be* sub-ordered *with respect to R, if for all distinct elements $m_1, m_2, \ldots, m_t$ in $\mathcal{M}$:*

(i) m_1 R m_1 (reflexivity), and

(ii) $\neg(m_1$ R m_2 and m_2 R $m_1)$ (antisymmetry), and

(iii) there does not exist a cycle between $m_1, m_2, \ldots, m_t$ (acyclicity).

If also (iv) (m_1 R m_2 or m_2 R m_1) $\forall\, m_1, m_2 \in \mathcal{M}$ (totality), then m_1 and m_2 are said to be comparable*, and $\mathcal{M}$ is* totally sub-ordered*.*

Furthermore, $\mathcal{M}$ is totally ordered *if R is reflexive, antisymmetric, total, and transitive, that is, (v) (m_1 R m_2 and m_2 R m_3) $\Rightarrow$ m_1 R m_3.*

One can prove that condition (i) and (iii) of Definition 3.10 are sufficient to define a sub-ordered set; nevertheless, we have listed all the three properties of sub-ordered sets for clarity and completeness.

Lemma 3.1. *A set $\mathcal{M}$ is totally ordered under a binary relation R if and only if it is totally sub-ordered.*

Proof. It is straightforward that transitivity and antisymmetry, together, imply acyclicity, i.e., if $\mathcal{M}$ is totally ordered, then $\mathcal{M}$ is also totally sub-ordered. Consider a chain of distinct elements in $\mathcal{M}$: m_1 R m_2, m_2 R $m_3, \ldots$, m_{t-1} R m_t. It follows that m_1 R m_t for the transitivity, and that m_t R m_1 cannot hold since R is antisymmetric.

We can also easily check that acyclicity and totality, together, imply transitivity. Consider the same chain of elements as above. Since for the acyclicity m_t R m_1 cannot hold, then, for the totality, m_1 R m_t. Therefore the transitivity holds, and the definitions of totally sub-ordered and totally ordered coincide. □

Now, before showing that some of these properties hold, we need to prove two preliminary results for the motif priority rule. At first, we observe that:

Fact 3.3. *The binary relation of motif priority is reflexive and antisymmetric, since properties (1), (2), and (3) of Definition 3.8 are strictly defined.*

Lemma 3.2. *(Paired occurrences) Let m and m' be two distinct motifs with $\min\{l \mid l \in \mathcal{L}_m \setminus \mathcal{L}_{m'}\} < \min\{l \mid l \in \mathcal{L}_{m'} \setminus \mathcal{L}_m\}$, such that both minima exist, and define j to be $\min\{l \mid l \in \mathcal{L}_m \setminus \mathcal{L}_{m'}\}$. Then, the occurrences of m and m' at positions less than j must be the same. Furthermore, we call these occurrences* paired.

Proof. We have to prove that, in case (3) of Definition 3.8, the occurrences of m and m' less than j are identical.

If m has an occurrence less than j that is not in $\mathcal{L}_{m'}$, then $\min\{l \mid l \in \mathcal{L}_m \setminus \mathcal{L}_{m'}\} < j$, which is impossible by hypothesis. Conversely, if m' has occurrences less than j that are not in $\mathcal{L}_m$, then $\min\{l \mid l \in \mathcal{L}_{m'} \setminus \mathcal{L}_m\} < j = \min\{l \mid l \in \mathcal{L}_m \setminus \mathcal{L}_{m'}\}$ contradicts again our assumptions. Thus, the occurrences of the two motifs less than j must coincide, and we call them paired.

Finally, by assumptions, it trivially holds that $j \notin \mathcal{L}_{m'}$. □

Lemma 3.3. *Let $m_1 \to m_2$, $m_2 \to m_3, \ldots, m_{t-1} \to m_t$ be a chain of distinct motifs of equal length and composition. Then, either $m_1 \to m_t$ or $\mathcal{L}_{m_t} \subset \mathcal{L}_{m_1}$ holds.*

Proof. We will prove the statement by induction on t. Let the basis be $t = 2$. In this case, the chain $m_1 \to m_2$ coincides with the result. We show now that, for $t > 2$, if it holds either $m_1 \to m_{t-1}$ or $\mathcal{L}_{m_{t-1}} \subset \mathcal{L}_{m_1}$, then either $m_1 \to m_t$ or $\mathcal{L}_{m_t} \subset \mathcal{L}_{m_1}$ holds.

Assume $\mathcal{L}_{m_{t-1}} \subset \mathcal{L}_{m_1}$. Define j to be $\min\{l \mid l \in \mathcal{L}_{m_{t-1}} \setminus \mathcal{L}_{m_t}\}$, and observe that it exists since $m_{t-2} \to m_{t-1}$, by property (3) of Definition 3.8. It follows that $j \in \mathcal{L}_{m_1}$ and, from Lemma 3.2, that the occurrences of m_{t-1} and m_t that are less than j are paired. Hence, these occurrences are also shared with m_1. Since $j \notin \mathcal{L}_{m_t}$, then either $m_1 \to m_t$, because j makes the difference between $\mathcal{L}_{m_1}$ and $\mathcal{L}_{m_t}$, or m_1 and m_t are incomparable. In the latter case, $\mathcal{L}_{m_{t-1}} \subset \mathcal{L}_{m_1}$ and $\mathcal{L}_{m_1} \subseteq \mathcal{L}_{m_t}$, together, imply that $\mathcal{L}_{m_{t-1}} \subset \mathcal{L}_{mt}$; that is impossible since m_{t-1} and m_t are comparable by hypothesis. This means that the latter case must be $\mathcal{L}_{m_t} \subset \mathcal{L}_{m_1}$.

Conversely, assume $m_1 \to m_{t-1}$. Define j as above, and $j' = \min\{l \mid l \in \mathcal{L}_{m_1} \setminus \mathcal{L}_{m_{t-1}}\}$. It holds that $j \notin \mathcal{L}_{m_t}$, as already observed, and that $j' \neq j$ because of $j' \notin \mathcal{L}_{m_{t-1}}$. Then, by Lemma 3.2, the occurrences of m_{t-1} and m_t that are less than j, are paired, and the same it is for occurrences of m_1 and m_{t-1} less than j'. In this way, if m_1 and m_t are comparable, then there exists an occurrence in $\mathcal{L}_{m_1}$ that appears first in s with respect to $\mathcal{L}_{m_t}$: if $j' < j$, the occurrences of m_1, m_{t-1}, and m_t less than j' are paired together, and thus j' makes the difference; otherwise $j' > j$, and the occurrences of m_1, m_{t-1}, and m_t less than j are paired together, and hence $j \in \mathcal{L}_1$ makes the difference. Alternatively, if m_1 and m_t are incomparable, then it must be the case of $\mathcal{L}_{m_t} \subset \mathcal{L}_{m_1}$. Otherwise, $\mathcal{L}_{m_1} \subseteq \mathcal{L}_{m_t}$ would imply that the occurrences of m_1 equal to or less than j' are also shared with m_t, which leads to $m_t \to m_{t-1}$; that is impossible. □

Theorem 3.1. *Any set of motifs $\mathcal{M}$ is sub-ordered with respect to the binary relation of motif priority.*

Proof. We have to prove that the relation of motif priority is reflexive, antisymmetric, and acyclic. The first two properties are stated in Fact 3.3. Now, following the work of Lemma 3.3, we can prove that the acyclicity holds too. First, observe that length and composition are intrinsic properties of the single motif, and thus monotone functions. If all motifs $m, m' \in \mathcal{M}$ have different length or composition, then, by definition of motif priority, it is always true that either $m \to m'$ or $m' \to m$ holds, and a cycle can never exist because of different lengths and/or compositions.

Alternatively, consider a chain of distinct motifs $m_1 \to m_2$, $m_2 \to m_3, \ldots$, $m_{t-1} \to m_t$ of equal length and composition. In this case we must use property (3) of Definition 3.8 to compare the motifs together. From Lemma 3.3, it follows that a cycle of motif priority between any chain of distinct motifs is again impossible, and therefore the acyclicity holds. Furthermore, since there may exist a triad of motifs m_1, m_2, m_3 of equal length and composition, such that $m_1 \to m_2$ and $m_2 \to m_3$, but $\mathcal{L}_{m_3} \subset \mathcal{L}_{m_1}$, then the motif priority rule is definitely not transitive on $\mathcal{M}$. □

Note that the non-decision on some binary comparisons finally discards the relations of totality and, as seen above, of transitivity. For instance,

consider s = AABEADBACE, C_1 = {A,C,D}, C_2 = {A,B,E}, and the motifs m = ([A,D][A,B][A,E], {2, 6}) and m' = ([A,D][B,E][A,E], {2, 6}) with the same list of occurrences: in short, $|m| = |m'| = 3$, $c(m) = c(m') = 6$, and $\mathcal{L}_m = \mathcal{L}_{m'}$. This means that we are not able to compare m and m' using the motif priority rule. Another example is given by m_1 = (A[A,C,D][B,E], {1, 5, 8}), m_2 = ([B,E]A[A,C,D], {4, 7}), and m_3 = ([A,B][C,D][B,E], {5, 8}). In this case, $m_1 \to m_2$ and $m_2 \to m_3$, but $m_1 \to m_3$ does not hold, since $\mathcal{L}_{m_3} \subset \mathcal{L}_{m_1}$. The issue is that m and m' may not be minimal motifs. In the following we set the basis to solve this problem.

Theorem 3.2. *Given any set of motifs $\mathcal{M}$, its minimal representative set $\mu(\mathcal{M})$ is totally ordered under the binary relation of motif priority.*

Proof. Along with Theorem 3.1 we have that any set of motifs $\mathcal{M}$ is sub-ordered under the motif priority rule; thus, also the set of minimal motifs $\mu(\mathcal{M})$ is sub-ordered. Following the work of Lemma 3.1, we have to prove that the totality holds on this new set $\mu(\mathcal{M})$, i.e., every pair of minimal motifs must be comparable under motif priority. In other words, if $m, m' \in \mu(\mathcal{M})$, either $m \to m'$ or $m' \to m$ must hold. To this end the proof of Lemma 3.1 tells us that length and composition are intrinsic properties of motifs, and thus every motif is comparable with another one in case of different length or composition.

From Remark 3.2 we have that $\mathcal{L}_m \neq \mathcal{L}_{m'}$ for two minimal motifs m and m' of equal length, and therefore there are, without loss of generality, two scenarios to consider: $\mathcal{L}_m \nsubseteq \mathcal{L}_{m'}$ with $\mathcal{L}_{m'} \nsubseteq \mathcal{L}_m$, and $\mathcal{L}_m \subset \mathcal{L}_{m'}$. In the former case, if we consider $\min\{l \mid l \in \mathcal{L}_m \setminus \mathcal{L}_{m'}\}$ and $\min\{l \mid l \in \mathcal{L}_{m'} \setminus \mathcal{L}_m\}$, then both the minima exist and are different from each other. Hence either $m \to m'$ or $m' \to m$ holds if it is the case of equal length and composition of m and m', and the minimum between the two sets is either in $\mathcal{L}_m$ or in $\mathcal{L}_{m'}$, respectively. Conversely, in the latter case, since m and m' are minimal motifs and the respective location lists are complete, the respective patterns of m and m' must be different. Therefore $\mathcal{L}_m \subset \mathcal{L}_{m'} \Rightarrow c(m) < c(m')$, and hence $m \to m'$. Accordingly, for any set of minimal motifs under motif priority, the totality holds. From Lemma 3.1 we can conclude that any set of minimal motifs is totally ordered under the motif priority rule. □

As a consequence of Theorem 3.2, all minimal motifs can be compared and ranked. We can further observe that, for property (2) of motif priority, every minimal motif has priority over all other motifs in $\mathcal{M}$ within its equivalence class, which is given by the paradigm of minimal representation.

Now it is clear that any set of motifs $\mathcal{M}$ can be mapped into its minimal representative set, $\mu(\mathcal{M})$, and that we can build a measure of total ordering over this set, in particular using the motif priority rule, that otherwise would not be possible on the original set $\mathcal{M}$.

3.3 Filtering by means of Underlying Motifs

In order to exploit the output of a motif discovery algorithm, our prior knowledge $\mathcal{M}$, and enhance its readability and interpretation, we aim to identify particular representative motifs of $\mathcal{M}$ that have to be $O(n)$ in number and dimensions, where n is the length of the reference string s.

Let R be a binary relation of priority between motifs. The objective is to select the most priority motifs in $\mathcal{M}$ for each location of s, according to R. If a motif m is selected, we discard all motifs with less priority that lie on the occurrences of m. The problem can be seen as a regional problem, in which we have to select some motifs for each considered region of the string s. These regions could be transcription factor binding sites or coding sequences if we consider genomes, otherwise functional regions if we consider a set of protein sequences. More formally:

Definition 3.11. *(Region) A* region $E_{j,x}$ *of s is a set of x consecutive locations $\{j, j+1, \ldots, j+x-1\}$ corresponding to the symbols $s_j s_{j+1} \ldots s_{j+x-1}$, namely a substring of s, where $1 \leq j$ and $(j+x-1) \leq |s|$.*

Definition 3.12. *(Living in a region) We say that a motif m of length k* lives *in the region $E_{j,x}$ if there exists a location $l \in \mathcal{L}_m$ such that $(E_{l,k} \cap E_{j,x}) \neq \emptyset$.*

Furthermore, we say that every motif m completely lives *in the regions defined by its occurrences.*

Definition 3.13. *(Tied occurrence) Let m be a motif of length k. Then, we say that an occurrence l of m is* tied *to a motif m', if m' lives in $E_{l,k}$ and m' R m. Otherwise, we say that l is* untied *from m'.*

Definition 3.14. *(Underlying representative set, Underlying motif) Let $\mathcal{M}$ be a set of motifs that lie on the string s, and let u be a positive integer called* underlying quorum. *A set of motifs $\mathcal{U} \subseteq \mathcal{M}$ is said to be an* underlying representative set *of s if and only if:*

(i) every motif m in $\mathcal{U}$, called underlying motif, *has at least u untied occurrences from any other motif in $\mathcal{U}$, and*

(ii) there does not exist a motif $m \in \mathcal{M} \setminus \mathcal{U}$ such that m has at least u untied occurrences from all motifs in $\mathcal{U}$.

In other words, given a string s and the information about all considered motifs $\mathcal{M}$, an underlying motif is a particular representative of some regions of s. Furthermore, considering the underlying quorum u as a fixed integer, it follows that Definition 3.14 of underlying representative set converges to one and only one set of motifs $\mathcal{U}$, under certain conditions:

Theorem 3.3. *Let $\mathcal{M}$ be a sub-ordered set of motifs with respect to a binary relation R. Then, there exists an underlying representative set $\mathcal{U} \subseteq \mathcal{M}$, and it is unique.*

Proof. We show first that a set $\mathcal{U}$, absolving the two conditions of Definition 3.14, exists. If $\mathcal{M}$ is sub-ordered, then there does not exist a cycle of priority between some distinct motifs $m_1, m_2, \ldots, m_t$ in $\mathcal{M}$ (acyclicity). Consider, without loss of generality, $m_1\ R\ m_t$, $m_2\ R\ m_t, \ldots$, $m_{t-1}\ R\ m_t$, such that no other motif has priority over m_t. This means that, for each region $E_{l,k}$ of s where m_t completely lives, either m_t or some other motifs in $\{m_1, m_2, \ldots, m_{t-1}\}$ can have an untied occurrence in $E_{l,k}$; thus they must belong to some set of underlying motifs, say $\mathcal{U}$. Furthermore, any other motif m_{t+1} living in $E_{l,k}$, but with no priority relation with respect to m_t, can also be an underlying motif in $\mathcal{U}$. Similarly, the occurrences of $\{m_1, m_2, \ldots, m_{t-1}\}$ living in $E_{l,k}$ must respect that rule. Since no cycle of priority is admitted, then, for all regions of s given by the occurrences of some motif m in $\mathcal{M}$, either m is in $\mathcal{U}$ or there are some other motifs in $\mathcal{U}$ that have untied occurrences from m and that cover those regions. In conclusion, there must be a set of underlying motifs $\mathcal{U}$.

In the opposite direction, we can prove that $\mathcal{U}$ is unique. Let us assume that there exist two distinct underlying representative sets $\mathcal{U}_1$ and $\mathcal{U}_2$. Then, for both sets, there exists at least one motif that is not in the other set, otherwise condition (ii) of Definition 3.14 does not hold. Suppose $m_1 \in \mathcal{U}_1 \setminus \mathcal{U}_2$. Since $m_1 \notin \mathcal{U}_2$, then, again for condition (ii), there must be an occurrence of m_1 in $\mathcal{U}_1$, say the region E_{l_1,k_1}, that in $\mathcal{U}_2$ is covered by some other motif m_2 with higher priority than m_1, such that $m_2 \notin \mathcal{U}_1$. For similar reasons, if m_2 is not in $\mathcal{U}_1$, then there must be another motif $m_3 \in \mathcal{U}_1$ with higher priority than m_2. Since $\mathcal{M}$ is sub-ordered, then $m_3 \neq m_1$. Finally, as the number of motifs in $\mathcal{U}_1$ and $\mathcal{U}_2$ is finite, then the two sets must coincide. □

The following result is a consequence of the proof of Theorem 3.3:

Proposition 3.1. *If $\mathcal{M}$ is totally ordered, then each underlying motif m has priority over all other underlying motifs in $\mathcal{U}$ in at least u regions of s.*

From this proposition, we have that, if R is a total relation over $\mathcal{M}$, then the number of underlying motifs on a string s is $O(n)$.

Theorem 3.4. *Let $\mathcal{M}$ be a totally ordered set under the binary relation R. Then, the number of underlying motifs in $\mathcal{U}$ is $\leq \lfloor n/2 \rfloor$, independently of the size of $\mathcal{M}$.*

Proof. From Proposition 3.1 we have that every untied occurrence does not overlap with the untied occurrences of some other motifs in $\mathcal{U}$, and thus for every motif m in $\mathcal{U}$ its untied occurrences will not overlap with those of other motifs. Since every motif m has length $k \geq 2$, then m must cover at least $2 + u - 1$ characters of s, obtaining at most $n/(2 + u - 1) \leq n/2$ possible different motifs in $\mathcal{U}$, independently of the size of $\mathcal{M}$. Furthermore, $|\mathcal{U}| \leq \lfloor n/(2u) \rfloor$ in case of non-overlapping occurrences for each of the motifs in $\mathcal{U}$. □

Let us now consider the *dimensions* of a set of motifs as the number of characters necessary to specify all its elements. This number coincides with the sum of compositions of all motifs. For instance, $m = (p, \mathcal{L}_m)$, with $p = a[a, c, d][b, e]$, is a motif with 6 characters and composition $c(m) = 6$.

Let r be the maximum number of characters allowed in a motif symbol, i.e., $r = \max_{1 \leq i \leq \tau}\{|C_i|\}$ as introduced in Section 3.2. Then:

Corollary 3.1. *Given a string s of length n and a totally ordered set $\mathcal{M}$, any underlying representative set $\mathcal{U}$ over $\mathcal{M}$ has a total number of symbols, or total length, $\leq \lfloor n/2 \rfloor$ and dimensions no greater than $|\Sigma|n$.*

Proof. Following the proof of Theorem 3.4, all underlying motifs in $\mathcal{U}$ can be placed on their untied occurrences in s without overlaps with other motifs. Therefore, the total number of symbols for the whole set of motifs $\mathcal{U}$ is $\sum_{m \in \mathcal{U}} |m| \leq \lfloor n/2 \rfloor$. Hence $\mathcal{U}$ has dimensions $\leq r\lfloor n/2 \rfloor \leq |\Sigma|n$. □

For example, consider a scheme that encodes every solid character on Σ plus the two square brackets '`[`' and '`]`', in a simple injective way. Then, the dimension of each encoding would be at most $\lceil \log(|\Sigma|+2) \rceil$. Therefore we can represent the symbols of a motif using at most $r\lceil \log(|\Sigma|+2) \rceil$ bits per symbol. If we encode every underlying motif with this scheme, then the overall space needed to store all motifs in $\mathcal{U}$ would be at most $r\lceil \log(|\Sigma|+2) \rceil \lfloor n/(2+u-1) \rfloor$ bits, or $r\lceil \log(|\Sigma|+2) \rceil \lfloor n/(2u) \rfloor$ bits in case of non-overlapping occurrences for each of the motifs in $\mathcal{U}$. This corresponds, in loose terms, to the result presented in Corollary 3.1. In conclusion, considering r as a constant, since it is bound by $|\Sigma|$, any underlying representative set $\mathcal{U}$ has linear dimensions with respect to the size of the string s.

To summarize the results of the previous sections, we have that every set of motifs $\mathcal{M}$, that could be the output of some motif discovery algorithm, can be transformed into its minimal representative set $\mu(\mathcal{M})$. This is the basis to enable the comparison of all motifs in $\mu(\mathcal{M})$ by means of the notion of motif priority, that captures the lexicographic power of a motif. One can prove the validity of a series of properties that are fundamental for the global comparison of motifs. On this setting we can define the subset of underlying motifs, that are, for some locations of the reference string s, the most important. In the next subsection we discuss an algorithm to find the set of underlying motifs, while in Section 3.4 we will describe a series of experiments to prove the validity of the latter.

3.3.1 The Proposed Algorithm

Here we describe an application of motif priority and underlying motifs. We present an algorithm that filters a set of motifs $\mathcal{M}$ lying on a string s into its underlying representative set $\mathcal{U}$.

Let R be a binary relation of priority. In the following we show how to build an underlying representative set $\mathcal{U}$ in case of a totally ordered set $\mathcal{M}$ under R. The purpose is to select the most representative motifs with character classes in $\mathcal{M}$, according to R and to the definition of underlying motif. In practice, we filter out those motifs in $\mathcal{M}$ that rank lower than others if located in the same regions of s. From Corollary 3.1, since $\mathcal{M}$ is totally ordered, it follows that $\mathcal{U}$ has linear dimensions.

Consider now the following algorithm for filtering the motifs in $\mathcal{M}$ into the underlying representative set $\mathcal{U} \subseteq \mathcal{M}$.

Algorithm 3.1.

1. Rank all motifs in $\mathcal{M}$ using the binary relation R.
2. Initialize $\mathcal{U}$ to an empty set, and iteratively select a motif m from $\mathcal{M}$ following the ranking given by the priority.
3. At each step, if m has at least u untied occurrences from all motifs already in $\mathcal{U}$:
4. (a) add m to $\mathcal{U}$ and store the regions of s in which m completely lives,
5. (b) otherwise, discard m.

The above algorithm is presented as a proof of concept. It is unclear if a more efficient implementation exists, or whether it is optimal. In this context we just analyze the correctness and the computational complexity of this algorithm. At first, we observe that the overall correctness follows from the proof of Theorem 3.3. Let n be the size of the string s, and let r be the maximum allowed size of a character class. If R corresponds to the motif priority rule, as of Subsection 3.2.2, in step 1 we may first preprocess the motifs in input, computing their lengths and compositions, in $O(rn|\mathcal{M}|)$ time. Then we sort all the motifs in $\mathcal{M}$, that are totally ordered, in descending

order. Considering the worst case, when we compare two motifs, we must also compare their lists of occurrences. Since for some motif m, $|\mathcal{L}_m|$ is $O(n)$, then the comparison of two ordered lists of occurrences costs $O(n)$ (see the pseudocode presented in the function CompareMotif). Thus, overall the first step costs $O(n|\mathcal{M}|\log|\mathcal{M}| + rn|\mathcal{M}|)$.

The main cycle, line 2, selects the top motifs from $\mathcal{M}$ and checks if they could be inserted in the set of underlying motifs $\mathcal{U}$. This loop can be stopped when no other motif can be added to the set $\mathcal{U}$.

After adding the first motif in the rank to $\mathcal{U}$, we check for the untied occurrences of the next motif m', having length k'. For each motif m that has been selected by our algorithm in $\mathcal{U}$, we store the occurrences of m in a vector of booleans $\Gamma[1,n]$, that represents all the locations of the string s. The value $\Gamma[i]$ is set to TRUE if the location i is tied to some motif m in $\mathcal{U}$. This means that, if m is in $\mathcal{U}$, then $\forall\, l \in \mathcal{L}_m$ we store the value TRUE at the locations of Γ that correspond to $E_{l,k}$, i.e., $\Gamma[l, l+k-1]$. Using the vector Γ, in the third instruction we have to check, for all $l' \in \mathcal{L}_{m'}$, whether some of the locations of $\Gamma[l', l'+k'-1]$, corresponding to the region $E_{l',k'}$, are set to TRUE. If it is the case, we say that l' is tied to some other motif with higher priority with respect to m'; otherwise, l' is untied. Accordingly, if m' has at least u untied occurrences, we add m' to the set $\mathcal{U}$; otherwise we discard m'. This step is detailed in the function CheckForUntiedOccurrences for a general binary relation R. The checking for untied occurrences of a motif m, using the additional vector Γ, thus costs $O(n)$.

In the fourth instruction we have to update the vector Γ with all locations of the newly added underlying motif m. This can be done by scanning all the occurrences of m, and, $\forall\, l \in \mathcal{L}_m$, updating the values $\Gamma[l, l+k-1]$ with no backtracking. To this end, we use a variable δ that takes the maximum between $l+k-1$ and the next occurrence l' of m. This procedure is described in the function StoreCoverage. Again this step takes $O(n)$ in time. In conclusion, the whole cycle of instructions 2–5 has a total cost of $O(n|\mathcal{M}|)$.

Since a motif m added to $\mathcal{U}$ has higher priority than the motifs considered at the iterations that follow m, as of the second instruction, then the following invariant holds: at each stage, condition (i) of Definition 3.14 is satisfied. Hence, from Theorem 3.3, $\mathcal{U}$ is the unique underlying representative set over

$\mathcal{M}$.

In summary, the total complexity of Algorithm 3.1 is dominated by the first term $O(n|\mathcal{M}|\log|\mathcal{M}| + rn|\mathcal{M}|)$. Moreover, if we consider r to be a constant as above, then the total complexity of the algorithm becomes $O(n|\mathcal{M}|\log|\mathcal{M}|)$. For completeness, below you can find the pseudocode of the described implementation of Algorithm 3.1, that we called MotifFiltering

Finally, in most cases we need first to compute the minimal representative set $\mu(\mathcal{M})$, before ranking the motifs by means of the priority rule. This further process costs $O(n^2|\mathcal{M}| + rn|\mathcal{M}|)$. Nevertheless, since we consider r as a constant, and typically the location lists of motifs have non-overlapping occurrences, that is a realistic assumption for most applications in genomics and proteomics, then in practice the complexity of computing $\mu(\mathcal{M})$ becomes $O(n|\mathcal{M}|)$. In this regard, filtering the minimal motifs in $\mu(\mathcal{M})$ ultimately takes $O(n|\mu(\mathcal{M})|\log|\mu(\mathcal{M})|)$ time.

MotifFiltering

Input: a sequence s, a list of motifs $\mathcal{M}$, an underlying quorum u, a binary relation R

Output: the underlying representative set $\mathcal{U}$

```
1  n ← length(s)
2  Γ ← a vector of n booleans set to FALSE, representing E_{1,n}
3  sort(M, R)
4  for each motif m = (p, L_m) in M
5       do test ← CheckForUntiedOccurrences(m, u, Γ)
6          if test is TRUE
7             then Γ ← StoreCoverage(m, Γ)
8                  U ← U ∪ {m}
9  return U
```

CheckForUntiedOccurrences

```
    Input: a motif m, an underlying quorum u, a vector of booleans Γ
    Output: the value TRUE, if the number of untied occurrences
    of m is at least u; the value FALSE, otherwise
 1  k ← length(m), delta ← 0
 2  for each occurrence l in L_m
 3        do count ← 0, untied ← TRUE
 4           delta ← max{delta, l}
 5           for each location i in E_{delta,l+k-1} such that
             (l + k − 1) ≤ |Γ|
 6                 do if Γ_i is TRUE
 7                       then untied ← FALSE
 8                            break for
 9           if untied is TRUE
10              then count + +, i + +
11                   if count ≥ u
12                      then return TRUE
13           delta ← i
14  return FALSE
```

StoreCoverage

```
    Input: a motif m, a vector of booleans Γ
    Output: the new vector of booleans Γ
1   k ← length(m), delta ← 0
2   for each location l in L_m
3         do delta ← max{delta, l}
4            store TRUE in the locations E_{delta,l+k-1} of Γ
5            delta ← l + k
6   return Γ
```

```
CompareMotifs
    Input: two minimal motifs m, m'
    Output: BEFORE, if m → m'; AFTER, if m' → m;
    EQUAL, otherwise
 1  if m = m'
 2     then return EQUAL
 3  if length(m) > length(m')
 4     then return BEFORE
 5  if length(m) < length(m')
 6     then return AFTER
 7  if c(m) < c(m')
 8     then return BEFORE
 9  if c(m) > c(m')
10     then return AFTER
11  ▷ At this stage, using the paired property described in
    Lemma 3.2, we can simply compare the two ordered
    lists of occurrences without any further check:
12  i ← 0
13  while TRUE
14        do if L_m.get(i) < L_m'.get(i)
15              then return BEFORE
16           if L_m.get(i) > L_m'.get(i)
17              then return AFTER
18           i++
19  return EQUAL
```

3.4 Experimental Results

In this section we discuss the ability of underlying motifs to efficiently capture meaningful information. A general problem in genome and proteome research is the identification of signals represented by means of motifs. In this sense, some modern degenerate motif discovery tools proved to be useful in biological sequence analysis, as we have discussed at the beginning of the chapter. Here we first collect the degenerate motifs in output from these

tools, say a set of motifs $\mathcal{M}$, and then present some experimental results in order to support the theoretical properties shown in the above sections.

More generally, there are two types of scenarios where the notion of underlying motifs could be useful. The first case is when a region of interest has already been identified, so that it is possible to analyze and select only those motifs that are underlying with respect to that particular region, without considering the whole set of motifs. Another possible application is the case where we just want to filter all motifs in $\mathcal{M}$, looking at the whole sequence.

In this context we present some results for the latter scenario. We take as input $\mathcal{M}$ the set of motifs extracted by Varun, a state-of-the-art tool for protein analysis. The benchmark consists of two protein families for which Varun successfully extracts the functional motifs [8], as reported in the PROSITE database. In particular, we consider the following protein families:

1. Nickel-dependent hydrogenases (id PS00508; in short, Ni). These are enzymes that catalyze the reversible activation of hydrogen, and are further involved in the binding of nickel. The family is composed by 22 sequences of about 12,300 amino acids in total.

2. Coagulation factors 5/8 type C domain (FA58C) (id PS01286; in short, Fa). This family is composed by 40 sequences of about 46,500 amino acids in total.

To prove that the information captured by the underlying motifs is relevant with respect to the protein families under examination, we devise two kinds of tests. In the first test we compare the original set of motifs in input, $\mathcal{M}$, with the set of underlying motifs in output $\mathcal{U}$, using global measures. In the second, we investigate the ability to filter a list of meaningful motifs and retain the functional ones, perhaps with a high rank.

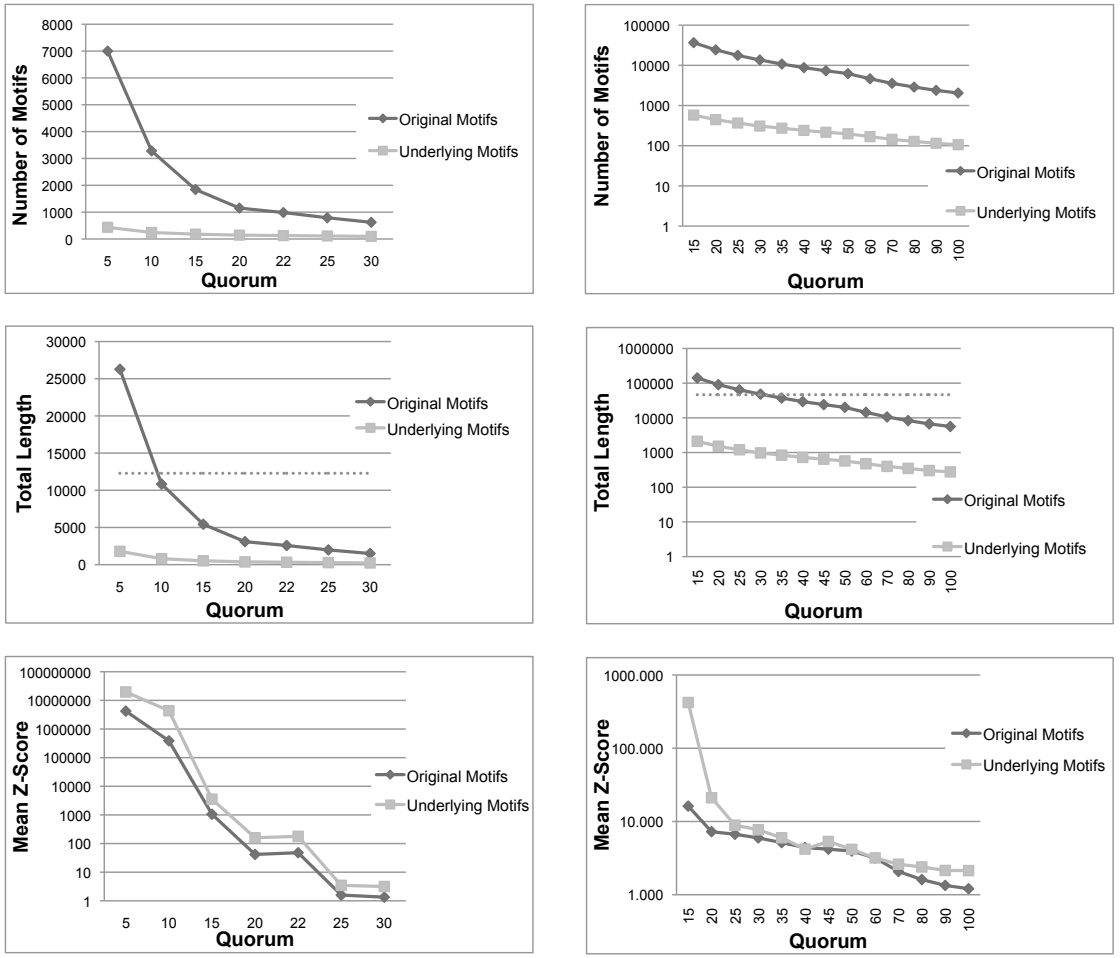
In the first set of experiments we use Varun to extract motifs from the above families of protein sequences for different quorums q. In all these experiments the quorum refers to the number of occurrences in the concatenation of all sequences of a certain family. Then, we use the extracted motifs as input to our algorithm, presented in Subsection 3.3.1, in order to compute the underlying motifs $\mathcal{U}$. Three observations must be made at this point:

- each motif $m \in \mathcal{M}$ extracted by Varun is with character classes and no gaps;
- before using Algorithm 3.1, we compute the set of minimal motifs $\mu(\mathcal{M})$;
- in our experiments we set the quorum u for the underlying motifs to be the same as that for the original motifs in input, thus $u = q$.

For both sets of motifs $\mathcal{M}$ and $\mathcal{U}$ we compute some statistics. Figure 3.3 shows the number of motifs, the sum of lengths of all motifs and their mean Z-Score. The Z-Score is computed employing the same formula reported in [8]. As expected, the number of underlying motifs is always much smaller than the number of the original motifs. A similar conclusion can be drawn for the sum of lengths. More important, for small quorums the total length of original motifs exceeds the length of sequences in both families, indicating that the original motifs are even larger than the set of sequences under examination. Moreover, as seen above, the sum of lengths of underlying motifs is always bounded by the length of sequences. These first two measures indicate that, not only the number, but also the total length of underlying motifs, is much smaller than the original motifs. Thus the filtering process is space-efficient.

Another important measure is the mean Z-Score of $\mathcal{M}$ and $\mathcal{U}$. The mean Z-Score is a global measure that captures the average quality of motifs in a set. In Figure 3.3 we can see that, for all quorums, the average Z-Score of underlying motifs is always greater than those of original motifs, and in most cases the difference is one or two order of magnitude. To summarize, this first test confirms that number and span of underlying motifs is much more manageable than the original set, and also that their average quality is improved.

Once we have verified that the notion of underlying is a suitable filter, in a second series of experiments we test the ability to retain meaningful motifs. To this end, we employed the candidate motifs obtained from the previous experiments to test the presence of functional motifs, as reported in the PROSITE database. In particular, the first family, Nickel-dependent hydrogenases, contains two functional motifs: $Ni_1 =$ `RG[FILMV]E`$\cdots\cdots\cdots\cdots$`[EM`



(a) Nickel-dependent hydrogenases (Ni) (b) Coagulation factors 5/8 type C domain (Fa)

FIGURE 3.3. *Total number, sum of lengths, and mean Z-Score of the motifs extracted using Varun and their corresponding underlying motifs, for the two protein families Ni and Fa.The dashed line in Total Length diagrams indicates the total size of each family. Note that in (a)–Mean Z-Score and (b)–All diagrams, the ordinate is plotted on a logarithmic scale.*

TABLE 3.3. PERFORMANCE OF THE UNDERLYING MOTIFS BASED ON MOTIF PRIORITY ON THE FAMILY Ni

Quorum	Max Sim. Ni_1 (underlying/original)	Max Sim. Ni_2 (underlying/original)
5	26/26	9/12
10	18/18	12/12
15	11/11	9/12
20	9/9	12/12
22	9/9	12/12
25	6/6	6/6
30	6/6	6/6

Normalized maximum similarity with the reference motifs of the family Nickel-dependent hydrogenases, for different quorums.

PQS][KR].C[GR][ILMV]C and Ni_2 = [FY]D[IP][CU][AILMV][AGS]C. Similarly, the second family, coagulation factors 5/8 type C domain (FA58C), shares the following functional motifs: Fa_1 = [FWY][ILV].[AFILV][DEGNST]$\cdot\cdot$ $\cdot\cdot\cdot\cdot$[FILV]$\cdot\cdot$[IV]$\cdot$[ILTV][KMQT]G and Fa_2 = [LM]R$\cdot$[EG][ILPV]$\cdot$GC. For both the sets of motifs, $\mathcal{M}$ and $\mathcal{U}$, we compute the maximum similarity between each motif in the set and the two functional motifs. The similarity between two motifs m and m' is the number of shared characters, including character classes, in the best alignment of m versus m', without considering indels. Tables 3.3 and 3.4 summarize the maximum similarity, for different quorums, of $\mathcal{M}$ and $\mathcal{U}$ with each functional motif of the two protein families Ni and Fa. The second and third columns report the maximum similarity for the set of underlying motifs divided by the similarity of the original set. For example, in the first row of Table 3.3 the quorum is 5. In this case, the maximum similarity of $\mathcal{M}$ with the functional motif Ni_1 is 26. The same value is obtained also for the corresponding set of underlying motifs $\mathcal{U}$, thus indicating that the motif Ni_1 is retained with the same degree of accuracy.

The values presented in Table 3.3 and 3.4 confirm that in most cases the functional motifs, that were present in $\mathcal{M}$, are also selected in the set

TABLE 3.4. PERFORMANCE OF THE UNDERLYING MOTIFS BASED ON MOTIF PRIORITY ON THE FAMILY Fa

Quorum	Max Sim. Fa_1 (underlying/original)	Max Sim. Fa_2 (underlying/original)
15	11/12	11/12
20	11/12	12/12
25	12/12	8/10
30	10/12	8/10
35	10/12	9/10
40	10/12	8/8
45	12/12	8/8
50	12/12	8/8
60	10/10	8/8
70	10/10	8/8
80	9/10	8/8
90	9/10	8/8
100	9/10	8/8

Normalized maximum similarity with the reference motifs of the family Coagulation factors 5/8 type C domain, for different quorums.

of underlying motifs with a similar accuracy. Moreover, we compared our motif priority rule with other standard ranking methods in application to the underlying motifs as binary relations. Table 3.5 reports the scores for each measure, where a large maximum similarity with the two PROSITE functional motifs and a high rank are preferable. Since these measures may not give a total ordering of motifs, apart from our motif priority, whenever necessary we randomly select among the motifs that have the same rank.

We can easily see that the motif priority rule achieves the best scores among all methods in identifying the actual functional motifs. In addition, our motif priority ranks on average the reference motifs of Ni in the top 3 out of 2,239 candidate motifs in input, and those of Fa in the top 5 out of 10,842 motifs.

Table 3.5. Comparison with Standard Ranking Methods

Binary Relation	Sim. $Ni_{1,2}$	Rank $Ni_{1,2}$	Sim. $Fa_{1,2}$	Rank $Fa_{1,2}$
Motif priority	151/157	2.78	247/264	5.34
Z-Score	127/157	5.00	223/264	9.96
Probability	127/157	5.00	223/264	9.96
Probability no back.	127/157	5.00	223/264	9.96
Frequency	93/157	22.78	168/264	9.42
Inv. frequency	118/157	6.14	212/264	5.69
Lexic. order occ.	93/157	5.50	142/264	11.77

Comparison of performance between different binary relations applied to the underlying motifs: motif priority, Z-Score, probability with distribution based on the amino acid frequencies in s, probability with no background (i.e, each amino acid scores $1/20$*), frequency of motifs in s, inverse frequency, and the lexicographic order of occurrences. For each family, we summed up the maximum similarity with the two representative motifs for all quorums. Similarly, in the column rank we show the average rank of the closest candidates to these motifs.*

Ultimately, although a more comprehensive experimental setting is desirable, these preliminary experiments support the validity of the theoretical results presented in the previous sections, and also prove their effectiveness in protein analysis.

3.5 Discussion and Future Work

In this chapter we have studied motifs with character classes, introducing notions for the comparison and ranking of motifs. We have proved several theoretical results that support the validity of these fundamental properties. Most important, our motif priority rule, together with the notion of underlying motifs, proved to be valuable for the analysis of biological sequences, bounding the total length of degenerate motifs in output from one or more modern motif discovery tools. Experiments on protein families have shown very good performance as a filter to reduce the number of motifs in output,

while keeping and ranking in the top 5 the most important ones.

The simple idea behind motif priority can be further employed in frameworks that more accurately distinguish the syntactic properties of single motifs. For example, we can set the minimum number of solid symbols needed for a motif to be considered, and then rank all motifs using the priority rule. Conversely, we can extend this concept to also take into account for motifs with don't cares and extensible motifs —i.e., motifs with a variable number of don't cares per site, as we have seen in Figure 3.2,— perhaps bounding the number of character classes and/or don't cares with some ratio (see for example [47]). Indeed our measure makes available a general framework of total ordering of the elements, where other more sophisticated measures can take place.

In the next chapter we apply the notion of underlying motifs to the pairwise comparison of whole genomes. For each region of the sequences under consideration, we will select the motif with the highest priority that has untied occurrence in both the sequences.

Chapter 4

Whole-Genome Phylogeny by virtue of Unic Subwords

The understanding of the whole human genome and of other living systems, and the mechanisms behind replication and evolution of genomes are some of the major problems in genomics. Although most of the current methods in genome sequence analysis are based only on genetic and annotated regions, this could saturate the problem because of their limited size of information. In fact, recent evidence suggests that the evolutionary information is also carried by the non-genic regions, and in some cases we cannot even estimate a complete phylogeny by analyzing the genes shared by a clade of species. Accordingly, this chapter addresses the phylogeny reconstruction problem for different organisms, namely viruses, prokaryotes, and unicellular eukaryotes, utilizing whole-genome pairwise comparisons.

With the progress of modern sequencing technologies a number of complete genomes is now available. Traditional motif discovery tools cannot handle this massive amount of data, therefore the comparison of complete genomes can be carried out only with ad hoc methods. In this work we propose a distance function based on paired-subword compositions, which extends the Average Common Subword approach (ACS) of Ulitsky *et al.* [113]. We first show that ACS is closely related to the cross entropy estimated between two entire genome sequences, and thus to some set of "independent" subwords, namely the irredundant common subwords. Then, our function

efficiently associates the irredundant common subwords to each region of the sequences under consideration, in order to remove certain repetitions inherent this notion of motifs. We thus filter the irredundant common subwords by means of underlying-paired motifs, which relate to each other the regions of two genome sequences. We call the selected motifs, underlying-paired irredundant common subwords, or simply *unic subwords*. In this framework, we also propose an extension to incorporate inversions and complements shared by the sequences.

In the last part of the chapter we finally present some experimental results for the 2009 human pandemic Influenza A (H1N1), Archaea and Bacteria domains, and the eukaryotic genus Plasmodium. These preliminary results show the validity of our method, and suggest novel computational approaches for analyzing the evolution of genomes.

4.1 Background

4.1.1 Whole-Genome Sequence Analysis

The global spread of low-cost high-throughput sequencing technologies has made publicly available a number of complete genomes, and this number is still growing quite rapidly day by day [102]. In contrast, only few computational methods can really handle as input entire chromosomes, or entire genomes. Similarly, the global alignment of large genomes has become a prohibitive task even for supercomputers, hence simply infeasible. To overcome this recent obstacle, in the last ten years a variety of alignment-free methods have been proposed. In principle they are all based on counting procedures that characterize a sequence based on its constituents, e.g., k-mers [23; 105].

For example, Sims *et al.* recently applied the Feature Frequency Profiles method (FFP) presented in [105] to compute a whole-genome phylogeny of mammals [104] —i.e., large eukaryotic genomes, including humans,— and of bacteria [106]. These significant results achieved by Sims *et al.*, give a significant example of the use of k-mers in genome analysis. In brief, they first estimate the parameter k in order to compute a feature vector for each sequence; this vector is composed by the frequency of each possible k-mer. In

general, once fixed k, the k-mers are 4^k in total for DNA, but a shorter vector is possible in case of reduced DNA alphabets. Each feature vector is then normalized by the total number of k-mers found (i.e., by the sequence length), obtaining a probability distribution vector, or feature frequency profile, for each genome. FFP finally computes the distance matrix between all pairs of genomes by applying the Jensen-Shannon divergence to their frequency profiles. For completeness, we notice that, in large eukaryotes, they filter out high-frequency and low-complexity features among all the k-mers found. Furthermore, in case the genomes have large differences in length, they use first the block method, similarly to [124], by dividing the sequences into blocks having the size of the smallest genome (with possible overlaps between the blocks). For each pairwise comparison, they finally average the minimum distance score between a block of a sequence versus all the blocks of the other sequence.

This general characterization of strings based on their subsequence composition closely resembles some of the information theory problems, and is tightly related with the compression of strings. In fact, compositional methods can be viewed as the reinterpretation of data compression methods, well known in the literature. For a comprehensive survey on the importance and implications of data compression methods in computational biology, we refer the reader to [44].

When comparing genomes it is well known that different evolutionary mechanisms can take place. In this framework, two closely related species are expected to share larger portions of DNA than two distant ones, whereby also other large complements and reverse-complements, or inversions, may occur [15; 61]. In this work we will take into account all these symmetries, in order to define a measure of comparison between genomes. We are interested in exploiting the system of large symmetries exhibited by the DNA molecules, toward the construction of a distance between whole genomes that incorporates inversions, duplications, and other simple genome rearrangements.

In this sense, an important fact is that most methods in the literature use only a portion of complete genomes [113]. For instance, there are approaches that use only the genic regions [23; 125] or the mitochondria [11; 73]; in other cases, methods filter out regions that are highly repetitive or with low

complexity, as for [105]. Recently, it has been shown that the evolutionary information is also carried by the non-genic regions [104]. For several families of viruses, we are not even able to estimate a complete phylogeny by analyzing their genes, since these organisms may share a very limited genetic material [113]. To avoid the possibility of filtering out informative regions, in our experiments we take entire genomes without any preprocessing. Note that a complete genome can be viewed as the concatenation of all its chromosomes, or segments for viruses, in a single string.

To address these all issues, our approach must pay special attention to the computational efficiency, and consequently to time and space complexities. In fact, even one of the most efficient tools for sequence alignment and comparison, MUMmer [67], fails and runs out of memory when large completely sequenced genomes are used.

4.1.2 Average Common Subword Approach

Among the many distance measures proposed in the literature, which in most cases are dealing with k-mers, as seen above, an effective and particularly elegant method is the Average Common Subword approach (ACS), introduced by Ulitsky *et al.* [113]. In short, given two sequences s_1 and s_2, where s_1 is the reference sequence, it counts the length $l[i]$ of the longest subword starting at position i of s_1 that is also a subword of s_2, for every possible position i of s_1 (see Table 4.1). This count is then averaged over the length of s_1. The general form of ACS follows:

$$ACS(s_1, s_2) = \frac{\sum_{i=1}^{|s_1|} l[i]}{|s_1|}.$$

The ACS measure is intrinsically asymmetric, but with simple operations can be reduced to a distance-like measure. In Subsection 4.2.5 we will give further details on how to adjust this formula; for the moment, we notice the similarity with the cross entropy of two probability distributions P and Q:

$$H(P, Q) = -\sum_x p(x) \log q(x),$$

where $p(x) \log q(x)$ measures the number of bits needed to code an event x from P if a different coding scheme based on Q is used, averaged over all the

possible events x.

TABLE 4.1. EXAMPLE OF COUNTERS $l[i]$ FOR THE ACS APPROACH

$s_1[i]$	A	C	A	C	G	T	A	C
$l_1[i]$	2	1	4	3	3	3	2	1

$s_2[j]$	T	A	C	G	T	G	T	A
$l_2[j]$	3	4	3	2	1	3	2	1

Counters $l_1[i]$ and $l_2[j]$ for the computation of $ACS(s_1, s_2)$ and $ACS(s_2, s_1)$, respectively, where $s_1 =$ `ACACGTAC`, $s_2 =$ `TACGTGTA`, and $i, j = 1, \ldots, 8$.

The beauty of the ACS measure is that it is not based on fixed length subwords, but it can capture also variable length matches, in contrast to most methods that are based on fixed sets of k-mers. In fact, with the latter the choice of the parameter k is critical, and every method needs to estimate k from the data under examination, typically using empirical measurements [105]. This may, however, overfit the problem and lead to loss of valuable information. In this spirit, ACS performs a massive genome sequence analysis, without limiting the resolution of motifs that can be naturally captured from sequences. Moreover, it does not filter out any region of the genomes under consideration.

4.1.3 Kullback-Leibler Information Divergence

As a matter of fact, Burstein *et al.* [21] proved that ACS indeed mimics the cross entropy estimated between two large sequences, supposed to have been generated by a finite-state Markov process. In practice, this is closely related to the Kullback-Leibler information divergence, and represents the minimum number of bits needed to describe one string, given the other:

$$D_{KL}(P \parallel Q) = H(P, Q) - H(P).$$

For example, if we consider the Lempel-Ziv mutual compression of two strings [70; 127], it parses one string based on a dictionary from the second

string, and in a similar way this mechanism is exploited by ACS. Since we are analyzing genome-wide sequences, this, asymptotically, can be seen as a natural distance measure between Markovian distributions.

The Kullback-Leibler divergence was introduced by Kullback and Leibler in 1951 [66]. This is also known as relative entropy, information divergence, or information gain. Studied in detail by many authors, it is perhaps the most frequently used information-theoretic similarity measure [65]. The relative entropy is used to capture mutual information and differentiation between data, in the same way that the absolute entropy is employed in data compression frameworks. Given a source set of information, e.g., s_2, the relative entropy is the quantity of data required to reconstruct the target, in this case s_1. Thus, this is a strong theoretical basis for the ACS measure. A drawback is that the Kullback-Leibler measure does not obey some of the fundamental axioms a distance measure must satisfy. In particular, Kullback-Leibler is not symmetric and does not satisfy the triangle inequality.

Despite these difficulties, ACS proved to be useful for reconstructing whole-genome phylogenies of viruses, bacteria, and eukaryota, outperforming in most cases the state-of-the-art methods [25; 113]. Moreover, it is computationally less demanding than other notable phylogenomic inferences like maximum parsimony and maximum likelihood, or other Bayesian estimations of divergence/correlation between entire genomes, where the correct estimation and use of the probability is often infeasible for practical problems —even when merely relegated to the analysis of genes and annotated regions, e.g., [35]. Therefore, here we aim to characterize and improve the ACS method, filtering out motifs that might be not useful for a whole-genome phylogeny of different organisms. In particular, we want to discard common motifs occurring in regions covered by other more significant motifs, for example according to the motif priority rule introduced in Chapter 3.

4.2 Materials and Methods

In this section we propose a distance measure between entire genomes based on UNderlying-paired Irredundant Common subwords, or *unic subwords*; a concept that extends the ACS method, and that we are going to define in

the following. We first notice that this chapter focuses on subwords, also called substring motifs. Unlike the different concepts of pattern treated in the previous chapters, we consider subwords in order to meet the demand for efficiency in the analysis of entire genomes.

4.2.1 Irredundant Common Subwords

In the literature, the values $l[i]$ captured by the ACS approach are called the *matching statistics*, as described in detail in Gusfield *et al.* [48] p. 132. Here we aim to characterize the matching statistics with associated motifs, in order to identify which motifs are essential for the ACS measure.

We thus recall the definition of irredundant common motifs given in Chapter 2 p. 22 and show that, in case we consider a motif domain restricted to only subwords (i.e., without mismatches/don't cares), there exists a close correspondence between the irredundant common subwords and the matching statistics.

Let s_1 and s_2 be two genome sequences on the four-letter DNA alphabet $\Sigma = \{\texttt{A,C,G,T}\}$, of lengths m and n respectively; and let us consider the set of all common subwords between s_1 and s_2. In this case, both strings and subwords are defined over the alphabet Σ; as usual, the length of a string or subword x is defined as the number of its symbols and denoted by $|x|$.

The occurrence $i \in \mathcal{L}_w$ of a common subword w, in either s_1 or s_2, is said to be *right-maximal* if and only if there is no other common subword w' occurring at i, such that $|w'| > |w|$; i.e., w' would extend w by appending one or more symbols to the right at the occurrence i. Similarly, the occurrence i of w is *left-maximal* if and only if no common subword w' occurs at $i-d \geq 0$, with d a positive integer, such that $|w'| \geq |w| + d$. Then, the occurrence i of a subword w is not *covered* by other subwords if and only if it is both right- and left-maximal. We call this latter type of occurrences *exposed*.

Definition 4.1. *(Irredundant/Redundant common subword) A common subword w is* irredundant *if and only if at least an occurrence of w in s_1 or s_2 is not covered by other common subwords. A common subword that does not satisfy this condition is called a* redundant common subword.

As in the case with don't cares, we note that every irredundant common subword w is the result of some intersection of the two entire sequences, where each meet, in this case, corresponds to particular a set of subwords. That is, if we correlate the exposed occurrences of w in a sequence with all its occurrences in the other sequence, we cannot extended w to the right or to the left without falling into a mismatch. We further observe that the set of all irredundant common subwords $\mathcal{I}_{s_1,s_2}$ is a subset of the well-known linear set of *maximal common subwords*, defined as the common subwords for which the list of occurrences cannot be deduced by the list of a longer subword, possibly adding an offset d (see [5; 112] for a more complete treatment of this topic). Therefore, the number of irredundant common subwords is bounded by $m + n$.

Now, let us define the vector $L_{x,y}$ of length $|x|$ composed by the matching statistics $L_{x,y}[i] = l_x[i]$.

Proposition 4.1. *We can achieve the matching statistics L_{s_1,s_2} and L_{s_2,s_1} combining together all and only the irredundant common subwords of s_1 and s_2.*

Proof. To show that such vectors L_{s_1,s_2} and L_{s_2,s_1} can be achieved with the irredundant common subwords, we define a new vector of scores l_w for each subword w, where $l_w[j] = |w| - j + 1$ represents the length of each suffix j of w, with $j = 1, \ldots, |w|$. Then, for each subword w in $\mathcal{I}_{s_1,s_2}$ we superimpose the vector l_w on the exposed occurrences of w in s_1 and s_2. L_{s_1,s_2} and L_{s_2,s_1} are finally obtained as the maximum of these scores for each location of the sequences.

We emphasize first that any occurrence of a common subword of s_1 and s_2 either corresponds to or is extended by one exposed occurrence of $\mathcal{I}_{s_1,s_2}$. This means that, by using the algorithm described above, in L_{s_1,s_2} and L_{s_2,s_1} we account for the maximum common subword length starting at each location of s_1 and s_2, respectively, as by definition of matching statistics.

In the opposite direction, we have to prove that all the irredundant common subwords are necessary to compute the matching statistics. This follows from the fact that a common subword is irredundant only if it has occurrences that are both right- and left-maximal, i.e., the exposed occurrences. There-

fore, every irredundant common subword w will exactly correspond to the matching statistics related to the starting positions of its exposed occurrences. Since the number of exposed occurrences for each subword $w \in \mathcal{I}_{s_1,s_2}$ is greater than one, this concludes the second part of the proof. Thus, all the irredundant common subwords are necessary and sufficient to directly compute the matching statistics L_{s_1,s_2} and L_{s_1,s_2}. □

A straightforward result of Proposition 4.1 is that we can exploit the set of irredundant common subwords $\mathcal{I}_{s_1,s_2}$, along with their location lists, in order to estimate the cross entropy between two sequences s_1 and s_2. At a glance, by employing this notion and the exposed occurrences of $\mathcal{I}_{s_1,s_2}$, we might be able to directly compute a symmetric similarity score between s_1 and s_2, that was previously not possible with the ACS method, without repeating the subsequence decomposition for both L_{s_1,s_2} and L_{s_2,s_1} [113]. In practice, it turns out that fast algorithms for the matching statistics are available in the literature. Thus, we can use the proof of Proposition 4.1 to perform a reverse engineering of the matching statistics, in order to compute the irredundant common subwords. More details about this passage will be given in Subsection 4.2.3.

In summary, the notion of irredundant common subwords is useful to decompose the information given by ACS into several patterns, and then perform an additional filtering on the most representative common motifs for each region of the sequences s_1 and s_2.

4.2.2 *Unic Subwords*

When comparing entire genomes we must focus on a one-to-one relation between different regions of the sequences under examination, so that to catch motifs preserved during evolution. We also want to avoid that large non-coding regions, which by nature tend to be highly repetitive, may overcount the same subwords a multiple number of times, misleading the final similarity score. In fact, while analyzing massive genomes, the number of repeated motifs is very high, particularly in the non-genic regions. For instance, in our experiments the number of irredundant common subwords can easily reach $2(m+n)/\log_4(m+n)$ elements in many pairwise comparisons, where

m and n are the lengths of s_1 and s_2; and a very large number of overlaps between these subwords is present. Therefore we need to filter out part of this information, and select for each region of the sequences the "best" common subword, by some measure, that matches it.

A useful technique is that of employing the *underlying motifs*, as presented in Chapter 3. For example, let us consider a region $E_{j,x}$, where j is a location of a sequence, say s_1, and x is the length of the region. In case there are two common subwords w and w' between s_1 and s_2 that occur in $E_{j,x}$ with an overlap, we must choose which subword retain for that region. The intuition is to take into account all the untied occurrences of the underlying representative set $\mathcal{U}$ (see Chapter 3 p. 61), and to add together their scores, mimicking ACS, in order to compute a similarity/distance measure between s_1 and s_2. Thus, following the interesting experimental results obtained with the ACS approach, here we aim to select the irredundant common subwords that best fit each region of s_1 and s_2, employing a technique that we call *Unic Subword Approach* or, in short, USA. This technique is based on a simple pipeline. It first selects the irredundant common subwords —retaining all the occurrences for completeness, and not only the exposed ones,— and subsequently filters out the subwords that are not underlying motifs.

In this regard, we must recall the definition of motif priority and of underlying motif, adapted from Chapter 3 to the case of pairwise sequence comparison. We will take as input the irredundant common subwords and the underlying quorum $u = 2$. Let now w and w' be two distinct subwords. We say that w has priority over w', or $w \to w'$, if and only if either $|w| \geq |w'|$, or $|w| = |w'|$ and w has the lexicographic order of its occurrences lower than w'. This order of occurrences must be configured for one of the two possible concatenations of s_1 with s_2, e.g., $s_1 s_2$ and $s_2 s_1$, repeating the process for the other concatenation. In this case, every subword can be defined just by its length and one of its starting positions in the sequences, meaning that any set of subwords is totally ordered with respect to the priority rule. Moreover, we say that an occurrence l of w is *tied* to an occurrence l' of a subword w', if $(E_{l,k} \cap E_{l',k'}) \neq \emptyset$ and $w' \to w$, where k and k' are, respectively, the lengths of w and w'. Otherwise, we say that l is *untied* from l'. This is a slightly modified definition of untied occurrence with respect to that introduced in

Chapter 3, in order to distinguish tied and untied occurrences.

Now, let $s = s_1 s_2$ be the string obtained through the concatenation of s_1 with s_2, and let $\mathcal{I}_{s_1,s_2}$ be the set of irredundant common subwords that lie on s.

Definition 4.2. *(Underlying-paired representative set, Unic subword) A set of subwords $\mathcal{U}_{s_1,s_2} \subseteq \mathcal{I}_{s_1,s_2}$ is said to be the* underlying-paired representative set *of s if and only if:*

(i) every subword w in $\mathcal{U}_{s_1,s_2}$, called unic subword*, has at least two occurrences that are untied from all the untied occurrences of other subwords in $\mathcal{U}_{s_1,s_2} \setminus w$, one in s_1 and one in s_2, and*

(ii) there does not exist a subword $w \in \mathcal{I}_{s_1,s_2} \setminus \mathcal{U}_{s_1,s_2}$ such that w has at least two untied occurrences, one per sequence, from all the untied occurrences of subwords in $\mathcal{U}_{s_1,s_2}$.

As for the underlying motifs, it is easy to see that this set of unic subwords exists, and is unique for a concatenation s. A direct procedure to discover the whole set $\mathcal{U}_{s_1,s_2}$ can be obtained from Algorithm 3.1, storing only the untied occurrences found for each selected subword. Furthermore, from Corollary 3.1 we know that the untied occurrences of the unic subwords can be mapped into the sequences s_1 and s_2 without overlaps in case of distinct subwords, resulting in a total length linear in the size of the sequences. To solve the problem of overlapped untied occurrences for a single subword, that in practice never occurs except in particular regions of genomes (e.g., telomers), we use a decision framework that randomly selects the occurrences and retains only those that do no overlap with the occurrences we chose earlier.

As already experienced, we notice that the underlying-paired subword selection might lead to an asymmetric distance, since the lexicographic order of occurrences plays a role in our priority rule. In fact, the concatenation of s_1 with s_2, where the order of the operands is critical, may result in different unic subwords with respect to the concatenation of s_2 with s_1. Therefore we compute two sets of unic subwords: $\mathcal{U}_{s_1,s_2}$ for $s = s_1 s_2$ and $\mathcal{U}_{s_2,s_1}$ for $s = s_2 s_1$; we will map their untied occurrences into s_1 for $\mathcal{U}_{s_1,s_2}$, and into s_2 for $\mathcal{U}_{s_2,s_1}$, as we will see in Subsection 4.2.5. In the following we focus our

attention on the discovery of all unic subwords for the general case $\mathcal{U}_{s_1,s_2}$, where we consider a sequence of the type $s = s_1 s_2$ in order to compute the lexicographic order of occurrences for each common subword of s_1 and s_2.

4.2.3 Efficient Computation of the Unic Subwords

In this chapter we are much interested in providing a proof of concept on the use of unic subwords in genome sequence analysis. This subsection gives some insights on how to efficiently find the unic subwords, whereas a more complete algorithmic framework would be desirable.

Unlike the ACS method that can efficiently compute the matching statistics, the algorithm we will describe in the following requires a little more computation due to the filtering of the underlying-paired motifs. We first show how to compute the irredundant common subwords from the matching statistics, and then we present an approach for the selection of the unic subwords among these motifs by exploiting some algorithmic techniques.

Discovery of the Irredundant Common Subwords

Proposition 4.1 allows us to compute the irredundant common subwords from the matching statistics in a simple way, thus exploiting the fast algorithms proposed in [48; 113]. These algorithms use two different data structures, either the suffix tree or the suffix array, to find all possible right-maximal occurrences of common subwords between s_1 and s_2. One can then map the length of each right-maximal occurrence i of a subword w into L_{s_1,s_2} or L_{s_2,s_1}, since this corresponds to the length $l[i]$ in one of these two vectors.

For simplicity, here we use the suffix tree data structure proposed by Weiner in 1973 [120], in its generalized version. The generalized suffix tree T_{s_1,s_2} for s_1 and s_2 indexes and stores all $m+n$ suffixes of the two sequences in a compact trie, or Patricia trie, in order to carry out fast string operations and search. The edges of T_{s_1,s_2} are labeled with strings such that each suffix corresponds to exactly one path from the root to a leaf, and each internal node is a branching node. The leafs of T_{s_1,s_2} are labeled with the index of the corresponding suffix; to differentiate, we say that the leafs from s_1 are colored with the color c_1, and those from s_2 with c_2. Furthermore, we denote

with $\bar{w}$ the node that spells the subword w in the path from the root to the node itself. Fast algorithms permit us to compute T_{s_1,s_2} in linear time with respect to the original sequences (for finite-size alphabets), since the number of its internal nodes is bounded by $m+n$ and many relations among these nodes are present. For the rest of this section, we assume familiarity of the reader with generalized suffix trees and their basic properties.

The first step in computing the irredundant common subwords consists in making a depth-first traversal of all nodes of T_{s_1,s_2}, and coloring each internal node with the colors of its leaves. In this traversal, for each leaf i of T_{s_1,s_2}, we capture the closest ancestor of i having both the colors c_1 and c_2, say the node $\bar{w}$. Then, w is a common subword, and i is one of its right-maximal occurrences (in s_1 or in s_2); we select all subwords having at least one right-maximal occurrence. The resulting set of subwords, that is linear in the size of the sequences, $O(m+n)$, represents a superset of the irredundant common subwords, since their right-maximal occurrences could be not left-maximal. Thus, we map the length of each right-maximal occurrence i into L_{s_1,s_2} and L_{s_2,s_1}, and, using Proposition 4.1, we check in a second step which occurrences have length greater than or equal to the length stored in the location $i-1$ (for locations $i \geq 2$), so that to capture subword occurrences that are not covered by the occurrences of other subwords. These latter occurrences are also left-maximal, and we can finally retain all subwords that have at least an occurrence that is both right- and left-maximal, i.e, the set of irredundant common subwords $\mathcal{I}_{s_1,s_2}$. Note that, by employing the above technique, we are able to directly discover the irredundant common subwords from the matching statistics L_{s_1,s_2} and L_{s_2,s_1}, if this information is already available. In this case, however, we would need other steps in order to collect a single list of these subwords and their occurrences in the sequences.

Then, with simple passages we can link together all the irredundant common subwords in a new tree $T^{\mathcal{I}}_{s_1,s_2}$ shaped on T_{s_1,s_2}, by creating edges between consecutive nodes (representing the irredundant common subwords) in the depth-first traversal of T_{s_1,s_2}, and removing the unused nodes and edges, except the root. Furthermore, we assign to each node $\bar{w}$ in $T^{\mathcal{I}}_{s_1,s_2}$, representing a subword $w \in \mathcal{I}_{s_1,s_2}$, all the occurrences of w that do not fall into the trees subtended by the children of $\bar{w}$, and we concatenate them in a vector de-

noted $\mathcal{L}_w^{r-m}$. Since the number of occurrences is bounded by $m+n$, then this operation is performed in linear time and space, retaining all the occurrences of a subword $w \in \mathcal{I}_{s_1,s_2}$ either in its vector $\mathcal{L}_w^{r-m}$ or in the vector of a node in its subtree.

In general, the construction of the generalized suffix tree T_{s_1,s_2} and the subsequent extraction of the irredundant common subwords $\mathcal{I}_{s_1,s_2}$ can be completed in time and space linear in the size of sequences. Alternatively, one can build either two distinct suffix trees or two distinct suffix arrays (i.e., arrays of integers giving the index of all suffixes of a string in their lexicographic order) for the sequences s_1 and s_2, in order to compute L_{s_1,s_2} and L_{s_2,s_1} and then extract the irredundant common subwords, by manipulating the subwords induced by the two trees/arrays. In practice, [113] suggests to build a suffix array for each sequence, since the typical time and space required for the construction of a suffix tree of a string of size n, is much more than n, due to its large constants. On the other hand, a suffix array built without resorting to a suffix tree, in general, requires $O(n \log n)$ time in the construction phase —even if more sophisticated techniques supporting linear time constructions have been recently developed [58],— and $O(\log n)$ time in finding a common subword. These are bounds also in practice, and thus one can choose the suffix array data structure to achieve fast and space-efficient implementations in genome analysis. Nevertheless, here we will continue to use a tree data structure for simplicity.

Selection of the Unic Subwords

Once acquired the irredundant common subwords and the tree $T^{\mathcal{I}}_{s_1,s_2}$, composed by at most $m + n$ nodes, we filter out the subwords that are not underlying-paired for the case $s = s_1 s_2$, obtaining the set of unic subwords $\mathcal{U}_{s_1,s_2}$. As seen in Chapter 3, this process first requires to sort the subwords. Then, other two steps are required for each subword w: checking for the untied occurrences of w, and storing these occurrences. We recall that m is the length of the sequence s_1.

First step. The first step can be easily done in linear time, as follows. For all subwords we retrieve their lengths and first occurrences in s_1 by a depth-

first traversal of $T^{\mathcal{I}}_{s_1,s_2}$; this step can be directly performed in the extraction of the irredundant common subwords, and we explain in the following the simple passages that solve this step. The length of each subword w is already stored in its corresponding node $\bar{w}$ in the tree, which was computed during the construction of the suffix tree. With a depth-first traversal of $T^{\mathcal{I}}_{s_1,s_2}$, we store, for each node, the smallest occurrence among the first occurrences in s_1 of its children. Moreover, we notice that length and first occurrence in s_1 uniquely characterize every possible subword.

Then, we map each (pointer to a) subword w into a vector of n boxes, according to the first occurrence of w in s_1. Each box i of the vector will contain all subwords that have i as first occurrence in s_1, and no pair of subwords in the box will have the same length. We further read this vector from the left, and map each subword w inside the boxes into a new vector of n queues, this time according to the length of w. We finally read the new vector from the left to achieve a ranking of all subwords according to the motif priority rule.

Second step. Let us consider two vectors Γ_1 and Γ_2 of m and n booleans, respectively, storing the locations of s_1 of s_2 covered by untied occurrences; if it is the case, a TRUE value is placed to cover a location. For simplicity, for each occurrence i of a subword we consider the related vector Γ, which is either Γ_1 if i belongs to s_1, or Γ_2 if i belongs to s_2.

Following the ranking, for each subword w under consideration we check for its untied occurrences from the list $\mathcal{L}^{r-m}_w$; this passage will not reduce the total occurrences of a subword, since we will add to $\mathcal{L}^{r-m}_w$ the occurrences of its children that could be untied for w, as we will see later. Then, for each occurrence i of w we need to check only its first and last location in the related vector Γ; i.e., we need to check the locations $\Gamma[i]$ and $\Gamma[i + |w| - 1]$, as follows by these simple observations:

- if one of these two values is set to TRUE, then i is tied to the occurrence of another subword w';
- otherwise, if both the values are set to FALSE, then i is untied from other subword occurrences. For example, consider a subword w', with

$|w'| \geq |w|$, scoring higher than w in the rank. If w' has an untied occurrence overlapping the region $E_{i,|w|}$, it would have set earlier $\Gamma[i]$ and/or $\Gamma[i + |w|]$ to TRUE.

If $\Gamma[i]$ is set to TRUE we completely discard the occurrence i, since it will be a tied occurrence also for the ancestors of $\bar{w}$. If both $\Gamma[i]$ and $\Gamma[i+|w|-1]$ are set to FALSE, we sign this occurrence to be stored as untied. However, in case w does not have in total at least one untied occurrence per sequence (i.e., one in Γ_1 and one in Γ_2), we do not store its untied occurrences and "send" all of them the parent node of $\bar{w}$, say $\bar{w}'$, by concatenation to $\mathcal{L}_{w'}^{r-m}$. In this way, i will be evaluate for w'.

If $\Gamma[i]$ is set to FALSE and $\Gamma[i + |w| - 1]$ is set to TRUE, we need to further evaluate this occurrence for the ancestors of $\bar{w}$. In this sense, one can easily compute the lower limit $\alpha = i + |w| - 1 - d$, with $d \geq 0$, below which $\Gamma[\alpha]$ is set to FALSE, for example by means of a length table in support (or in substitution) of the boolean vector Γ. A length table stores an untied occurrence i of a subword w in such a way that for its locations $[i, i+1, \ldots, i+|w|-1]$ we have the values $[1, 2, \ldots, |w|]$, respectively. Therefore we "send" the occurrence i to the closest ancestor of $\bar{w}$ that has length $< |w| - d$, say $\bar{w}''$, by concatenating i to $\mathcal{L}_{w''}^{r-m}$. This step can be performed by adapting classical algorithms for the level ancestor problem to the case of suffix trees. A simple algorithm that solves the level ancestor problem for non-compact tries is provided by Bender and Farach-Colton [19]; with a linear preprocessing of the tree, they can find a particular ancestor in constant time. In our case, we can find the suitable ancestor in time $O(\log\log\max\{m, n\})$, with $O(m + n)$ preprocessing of the entire tree $T_{s_1,s_2}^{\mathcal{I}}$. The $O(\log\log\max\{m, n\})$ bound comes from the predecessor search in weighted trees [62], such as suffix trees, where $\max\{m, n\}$ is the maximum possible height for $T_{s_1,s_2}^{\mathcal{I}}$.

At this point, one can easily see that each occurrence i is evaluated at most $O(\log\max\{m, n\})$ times, since there could be at most $O(\log\max\{m, n\})$ submissions to ancestor nodes. Suppose, for example, that i was originally belonging to $\mathcal{L}_w^{r-m}$ of a subword w. Thus, we first evaluate i with the subword w. Consider, without loss of generality, $\Gamma[i]$ set to FALSE and $\Gamma[i+|w|-1]$ set to TRUE, and that w has at least one other untied occurrence per sequence. In this case, we select w and send i to $\mathcal{L}_{w'}^{r-m}$ of a particular ancestor $\bar{w}'$ of $\bar{w}$,

clearly with $|w'| \leq |w|$. Again, when evaluating the subword w', if $\Gamma[i]$ is set to FALSE and $\Gamma[i+|w'|-1]$ is set to TRUE, this means that a subword w'' with $|w''| \geq |w'|$ has since covered some locations of the region $E_{i,\alpha-1}$ of Γ, where α is defined as above. Otherwise, we sign this occurrence as a (possible) untied occurrence for w'. Thus, for each evaluation of i not successful, say the iteration $j > 1$ corresponding to the ancestor $\bar{w}'$ of $\bar{w}$, we have since the iteration $j-1$ covered $E_{i,i+|w|-1}$ with a subword w'' of at least the size $|w|$, which further overlaps the region $E_{i,i+|w'|-1}$. This means that the worst case is when $|w'|$ and $|w''|$ have about half the size of the subword evaluated at the iteration $j-1$ for the occurrence i, since it must be $|w''| \geq |w'|$. For these reasons, at most $O(\log |w|)$ evaluations of i can be made.

Furthermore, to avoid boundaries effects, for each subword w one would distinguish its occurrences in s_1 and in s_2, and check first the size of these two sets, i.e., of $\mathcal{L}_w^{s_1-r-m}$ and $\mathcal{L}_w^{s_2-r-m}$. In case one of them is empty, we avoid to further check for untied occurrences in the other set, and send them the parent node of $\bar{w}$.

Third step. In this step we store the untied occurrences signed for each unic subword w in the vectors Γ_1 and Γ_2. In case w does not have at least an untied occurrence per sequence, we avoid to store its signed occurrences and send them to the parent of $\bar{w}$, in order to be evaluated again in a later iteration.

This global step clearly takes linear time in m, since the untied occurrences of distinct subwords do not overlap together by definition. In case there are overlaps between the untied occurrences of a single subword, we randomly select among these occurrence, storing those that do not overlap with the ones we chose earlier. Moreover, if we discard some of these latter occurrences because of overlaps, we follow for them the same procedure as for the other tied occurrences of w.

In conclusion, our approach requires $O((m+n)\log\max\{m,n\}\log\log\max\{m,n\})$ time and $O(m+n)$ space to discover the set of all unic subwords $\mathcal{U}_{s_1,s_2}$ by employing a generalized suffix tree for s_1 and s_2. We then need to repeat the approach for $\mathcal{U}_{s_2,s_1}$ in order to achieve a symmetrical score for the two sequences.

4.2.4 Extension of Our Approach to Inversions and Complements

As discussed above, one may further investigate the use of two single suffix arrays, one for each sequence, to enhance the computation of the irredundant common subwords, and subsequently the selection of the unic subwords. We chose a generalized suffix tree data structure to better show the global structure of our approach, that can be easily extended to account also for inverse and complement matches between s_1 and s_2.

An simple idea is to concatenate each sequence with its inverse and its complement. In this way we keep separate the occurrences coming from direct matches, inversions, and complements. In brief, we first define $\hat{x}$ as the concatenation of string x with its inverse, following by its complement, in this exact order. Then, we compute the irredundant common subwords on the sequences $\hat{s_1}$ and $\hat{s_2}$. We subsequently select the unic subwords by ranking all irredundant common subwords —where the lexicographic order of occurrences is based on the concatenation $\hat{s} = \hat{s_1}\hat{s_2}$, in the case of $\mathcal{U}_{s_1,s_2}$,— and then mapping each subword occurrence on the reference sequences s_1 and s_2. In case an occurrence refers either to the inverse or to the complement of the two sequences, we adjust its location with a proper shift. In this way we can store all the untied occurrences found on Γ_1 and Γ_2, which maintain the size of the original sequences, and consider all possible matches for each region of s_1 and s_2.

More computing and storage are necessary to analyze also the inverse and the complement of each sequence, while maintaining the asymptotic computational complexity and space. In our framework, we chose to take into account for all these symmetries, and thus the experiments we will present in the last part of the chapter reflect the use of this extended approach.

4.2.5 A Distance-like Measure based on Unic Subwords

In the following we report the basic steps of our distance-like measure, similarly to ACS.

Let us assume that we have computed $\mathcal{U}_{s_1,s_2}$, which refers to the concatenation $s = s_1 s_2$ —the other set, $\mathcal{U}_{s_2,s_1}$, will be used in the specular case $s = s_2 s_1$. For every subword $w \in \mathcal{U}_{s_1,s_2}$ of length k we sum up the score

$h_w^{s_1} \sum_{i=1}^{k} i = h_w^{s_1} k(k+1)/2$ in $USA(s_1, s_2)$, where $h_w^{s_1}$ is the number of its untied occurrences in s_1 with respect to $\mathcal{U}_{s_1,s_2}$ (i.e., those stored in Γ_1). Then, we average $USA(s_1, s_2)$ over the length of the first sequence, s_1, yielding

$$USA(s_1, s_2) = \frac{\sum_{w \in \mathcal{U}_{s_1,s_2}} h_w^{s_1} |w|(|w|+1)}{2|s_1|}.$$

This is a similarity score that is large when two sequences are similar, therefore we take its inverse. Moreover, for a fixed sequence s_1 this score can also grow with the length of s_2, since the probability of having a match for each region of s_1 increases with the length of s_2. For this reasons, we consider the measure $\log_4(|s_2|)/\ USA(s_1, s_2)$, where $\log_4(m)$ represents in general, in our analysis, the minimum length captured by the unic subwords by removing high-frequency subwords; and 4 is the alphabet size. Another issue of the above formula is the fact that it does not converge to zero for $s_1 = s_2$; thus we subtract the correction term $\log_4(|s_1|)/\ USA(s_1, s_1)$, which ensures that this condition is always satisfied. Since $\mathcal{U}_{s_1,s_1}$ contains only one subword, the sequence s_1 itself, which trivially has only one untied occurrence in s_1, this yields to $USA(s_1, s_1) = |s_1|(|s_1|+1)/(2|s_1|) = (|s_1|+1)/2$. The following formulas accommodate all of these observations in a symmetrical distance-like measure $d_{USA}(s_1, s_2)$ between the sequences s_1 and s_2:

$$\overline{USA}(s_1, s_2) = \frac{\log_4(|s_2|)}{USA(s_1, s_2)} - \frac{2\log_4(|s_1|)}{(|s_1|+1)},$$

$$d_{USA}(s_1, s_2) = \frac{\overline{USA}(s_1, s_2) + \overline{USA}(s_2, s_1)}{2}.$$

We can easily see that the correction term rapidly converges to zero as $|s_1|$ increases; then, for genome sequences over the alphabet Σ, $d_{USA}(s_1, s_2)$ is ≥ 0, and $d_{USA}(s_1, s_2) = 0$ if and only if $s_1 = s_2$. Moreover, we notice that $d_{USA}(s_1, s_2)$ grows as the two sequences s_1 and s_2 diverge.

From now we will simply refer to the measure $d_{USA}(s_1, s_2)$ as the Unic Subword Approach measure, or USA. As for the ACS approach, $d_{USA}(s_1, s_2)$ may not satisfy the triangle inequality. This, however, does not seem to be reflected in our experiments, where the triangle inequality holds in almost all tests carried out for both the approaches USA and ACS.

4.3 Experimental Results

4.3.1 Genome Datasets and Reference Taxonomies

We assess the effectiveness of the Unic Subword Approach on the estimation of whole-genome phylogenies of different organisms. We test our distance function on three types of datasets that consider complete genomes among viruses, prokaryotes, and unicellular eukaryotes.

In the first dataset we selected 54 virus isolates of the 2009 human pandemic Influenza A – subtype H1N1, also called the "Swine Flu," which had spread all over the world throughout the whole year 2009, including different flu seasons. Within the influenza A virion are eight segments of viral RNA with different functions; each RNA is copied into DNA in order to sequence the viral genome. We chose the sequences among those described in [103], retaining the isolates with all the 8 segments completely sequenced and equally distributed among the world geographic regions (see Figure 4.1 for a complete list of the viruses). All segments accession numbers can be found in the supplementary material of [103] or in the Influenza Research Database by typing their complete name;[1] the related nucleotide FASTA sequences can then be downloaded from the National Center for Biotechnology Information (NCBI)[2] of the U.S. National Institutes of Health. We concatenate these segments by means of a symbol not in Σ, e.g., '`$`' or '`N`', according to their natural order. The resulting sequences are very similar (in some cases almost identical), and have lengths in the order of 13,200 nucleotide base pairs (bp) each, accounting for a total of 714,402 bp. To compute a reference taxonomic tree, we perform an extensive multiple sequence alignment using the ClustalW2 tool version 2.1,[3] as suggested by many scientific articles on the 2009 Swine Flu [103; 108]. Then, we compute the tree using the DNAML tool from the PHYLIP software package[4] release 3.69 [38], which implements the maximum likelihood method for DNA sequences.

[1] The Influenza Research Database (FluDB) is available at `http://www.fludb.org`.

[2] The NCBI database is available online at `http://www.ncbi.nlm.nih.gov`.

[3] ClustalW2 is available at `http://www.ebi.ac.uk/Tools/msa/clustalw2`.

[4] PHYLIP (phylogenetic inference package) is a free computational phylogenetics software package available at `http://evolution.genetics.washington.edu/phylip`.

In the second dataset we selected 18 prokaryotic organisms among the species used in [113] for a prokaryotic DNA genome phylogenomic inference (see Table 4.2). We chose the species whose complete genome has been sequenced and published, and whose phylogenetic tree structure can be inferred with well-established methods in the literature. Table 4.2 highlights that the organisms come from both major prokaryotic domains: Bacteria, 10 organisms in total, and Archaea, 8 organisms in total. All genomes have been downloaded from the NCBI genome database.[5] The sequences in question have lengths ranging from 1 Mbps to 5 Mbps, accounting for a total 48 Mbp. We compute their tree-of-life by using genes that code for the 16S ribosomal RNA, a small ribosomal subunit characterizing prokaryotes and widely used to reconstruct their phylogeny [26]. These genes are referred to as 16S rDNA. We can extract a multiple alignment of 16S rDNA sequences of the selected organisms from the Ribosomal Database Project release 8.1.[6] We then perform a maximum likelihood estimation on the aligned set of sequences, and use DNAML from PHYLIP in order to compute a reference tree based on the resulting estimations.

In the third dataset we selected 5 eukaryotic taxa of the protozoan genus *Plasmodium* whose genomes have been completely sequenced. Plasmodium are unicellular eukaryotic parasites best known as the etiological agents of malaria infectious disease, one of the greatest threats to humankind. It is estimated that malaria kills a million people a year in the Sub-Saharan Africa, most of them are children under five and pregnant women. Table 4.3 lists names and global features of each selected parasite. The sequences have lengths ranging from 18 Mbp to 24 Mbp, accounting for a total 106 Mbp. In this case, we can extract the complete DNA genomes from the PlasmoDB database[7] [12] (in our tests we used the release 7.2), and concatenate each chromosome through a symbol not in Σ. We used as reference tree the taxonomy computed by Martinsen *et al.* [78], as suggested by the Tree of Life Web Project (ToL).[8]

[5]The NCBI genome database is available at `ftp://ftp.ncbi.nlm.nih.gov/genomes`.

[6]The Ribosomal Database Project is available at `http://rdp.cme.msu.edu`.

[7]PlasmoDB is available at `http://www.plasmodb.org`.

[8]The Tree of Life web project is hosted by the University of Arizona and available at `http://www.tolweb.org`.

TABLE 4.2. BENCHMARK FOR PROKARYOTES – ARCHAEA & BACTERIA DOMAINS

Accession No.	Domain	Organism	Size
BA000002	Archaea	Aeropyrum pernix str. K1	1.7 Mbp
AE000782	Archaea	Archaeoglobus fulgidus str. DSM 4304	2.2 Mbp
AE009439	Archaea	Methanopyrus kandleri str. AV19	1.7 Mbp
AE010299	Archaea	Methanosarcina acetivorans str. C2A	5.8 Mbp
AE009441	Archaea	Pyrobaculum aerophilum str. IM2	2.3 Mbp
AL096836	Archaea	Pyrococcus abyssi	1.8 Mbp
AE009950	Archaea	Pyrococcus furiosus str. DSM 3638	1.9 Mbp
AE000520	Archaea	Treponema pallidum sp. pall. str. Nichols	1.2 Mbp
AE017225	Bacteria	Bacillus anthracis str. Sterne	5.3 Mbp
AL009126	Bacteria	Bacillus subtilis subsp. subtilis str. 168	4.3 Mbp
AE013218	Bacteria	Buchnera aphidicola str. Sg	651 kbp
AL111168	Bacteria	Campylobacter jejuni sp. jej. str. NCTC 11168	1.7 Mbp
AE002160	Bacteria	Chlamydia muridarum str. MoPn/Wiess-Nigg	1.1 Mbp
AM884176	Bacteria	Chlamydia trachomatis str. L2/434/Bu	1.1 Mbp
AE016828	Bacteria	Coxiella burnetii str. RSA 493	2.0 Mbp
AE017285	Bacteria	Desulfovibrio vulgaris sp. vulg. str. Hildenb.	3.6 Mbp
L42023	Bacteria	Haemophilus influenzae str. Rd KW20	1.9 Mbp
CP001037	Bacteria	Nostoc punctiforme str. PCC 73102	8.4 Mbp

Prokaryotic taxa used in our experiments, divided by domain. For each entity, we list the accession number in the NCBI genome database, the complete name and strain, and the complete DNA genome size.

4.3.2 Whole-Genome Phylogeny Reconstruction

We exploited the above datasets to compare our method, the Unic Subword Approach (USA), with other efficient state-of-the-art approaches in the whole-genome phylogeny reconstruction challenge: ACS, FFP, and FFP_{RY}. The FFP_{RY} method, instead of FFP, employs a reduced alphabet, the Purine-Pyrimidine alphabet (RY), which is composed by two character classes: `[A,G]` (both purine bases, denoted by `R`) `[C,T]` (both pyrimidines, denoted by `Y`). We implemented the ACS method by ourselves, while for FFP and FFP_{RY}

TABLE 4.3. BENCHMARK FOR UNICELLULAR EUKARYOTES – GENUS PLASMODIUM

Parasite	Host	Region	Size
P. Berghei	Rodent	Africa	18.5 Mbp
P. Chabaudi	Rodent	Africa	18.8 Mbp
P. Falciparum	Human	Africa, Asia & S./C. America	23.3 Mbp
P. Knowlesi	Macaque	Southeast Asia	23.7 Mbp
P. Vivax	Human	Africa, Asia & S./C. America	22.6 Mbp

Eukaryotic organisms of the unicellular genus Plasmodium whose genome has been completely sequenced. Plasmodium are parasites known as causative agents of malaria in different hosts and geographic regions. The right-most column lists the size of each complete DNA genome.

we used the FFP package release 3.14 available online.[9] We did not implement the Block method for any of these algorithms, nor we performed other tuning operations on the methods or a preliminary filtering of the sequences.

For each benchmark, every method produces a distance-like matrix whose cells refer to the pairwise distance between different organisms, and we check that a zero value is placed at each location of the diagonal. We reconstruct the phylogenomic trees from the distance matrices using the Neighbor-Joining algorithm (NJ) of Saitou and Nei [99], as implemented by the NEIGHBOR tool in the PHYLIP package. By default this method returns binary unrooted trees whose leaves are the different entities in input; it further discards the branch lengths.

We compute the symmetric difference of Robinson and Foulds (R-F) [95] to compare the resulting topologies, assuming all edges of unit length, to the respective reference trees. In brief, we consider all possible branches on the two trees. Each of these edges partitions the leaves into two subsets, the leaves connected to one end of the branch and those connected to the other end. R-F counts the number of partitions that are not common to both trees. For two unrooted binary trees with $n \geq 3$ leaves, the R-F score

[9]The FFP software package release 3.14 is available at `http://ffp-phylogeny.sourceforge.net`.

is in the range $[0, 2n-6]$, since each tree has $2n-3$ edges, and is always even. The lower this value is, the closer the obtained tree to the reference tree: 0 means that the two trees are isomorphic, while $2n-6$ means that all non-trivial bipartitions are different —a trivial bipartition corresponds to an edge incident to a leaf, therefore there are n trivial bipartitions per tree, and $2(2n-3-n) = 2n-6$ non-trivial in total. This is a standard method to evaluate distance matrices for phylogenomic purposes. However, R-F suffers from the small possible variations that may occur between the two trees, resulting in a huge R-F score when dealing with a number of organisms (typically, already for $n \sim 10$). Moreover it does not take into account branch lengths, as seen above. We can finally compute the R-F difference between two or more trees using the TreeDist tool from the PHYLIP package.

4.3.3 Performance Comparison and Statistics

Table 4.4 compares the phylogenomic reconstruction of our method with that of the other state-of-the-art approaches, by showing the R-F difference with respect to the reference taxonomy of each species. The different types of datasets —viruses, prokaryotes, and unicellular eukaryotes— are also highlighted. We ran FFP and FFP_{RY} for different values of k (the fixed subword length) as suggested by [105], retaining the best results in agreement with the reference trees. In general, FFP works better with $k \sim \lceil \log_4 m \rceil$, where m is the maximum length of a sequence in a clade, and FFP_{RY} with $k \sim \lceil \log_2 m \rceil$. Our method, USA, obtains good performance in every test if we consider the R-F difference, and very good performance if we further analyze the phylogenies, as of figures 4.1, 4.2, and 4.3. We achieve in all cases at least the score of the best performing method, outperforming the other methods for sequences that share large parts, as in the case of viruses.

More in detail, Figure 4.1 shows that our approach can distinguish the two main clades of the 2009 Swine Flu (in green and red), whose have been outlined in [103] and in other major works on the Swine Flu. The origin of the flu could reside in the considered Mexican isolate of early April 2009 (Mexico/4108, in green), to which all other green isolates may ensue, from California/06 to the European isolates. Two sub-clades for the U.S. states

TABLE 4.4. COMPARISON OF WHOLE-GENOME PHYLOGENY RECONSTRUCTIONS

Species	Group	USA	ACS	FFP	FFP_{RY}
Influenza A	Viruses	80/102	84/102	100/102	96/102
Archaea	Prokaryotes	4/10	4/10	6/10	6/10
Bacteria	Prokaryotes	6/14	10/14	6/14	10/14
Arch. & Bact.	Prokaryotes	20/30	22/30	20/30	22/30
Plasmodium	Unic. Eukaryotes	0/4	0/4	4/4	0/4

Normalized Robinson-Foulds score of the agreement with the reference taxonomy for each reconstruction method and species.

of California and Texas are highlighted within the red clade, most probably corresponding to the first major evolutions of the viral disease, to which New York, Italy, Denmark, Russia, China, and Japan viruses may have developed from.

Figure 4.2 plots the whole-genome phylogeny of all prokaryotic organisms analyzed, computed by our method. USA can easily distinguish the Archaea domain, in red, from the Bacteria domain, in green, and also other sub-clades with respect to the reference tree (these sub-clades are highlighted in the figure with different colors). Also, the organisms in black does not form a clade with other organisms in the reference tree. In this case we set the length of branches to a unit value, in order to better visualize the tree.

Figure 4.3 finally illustrates the whole-genome phylogeny of the genus Plasmodium generated by USA, which corresponds completely to the taxonomy found in the literature. In this figure we also provide our whole-genome distance on the branches, after the application of the Neighbor-Joining algorithm.

In Table 4.5 we present some statistics for the unic subwords based on the experiments performed. First, we can see that only a few subwords are selected on average among the irredundant common subwords for each genome comparison. Removing the high-frequency subwords (which were very few), we notice that the unic subwords typically have lengths $\geq log_4 m$, and in the case of viruses they can also be very large, capturing more information than

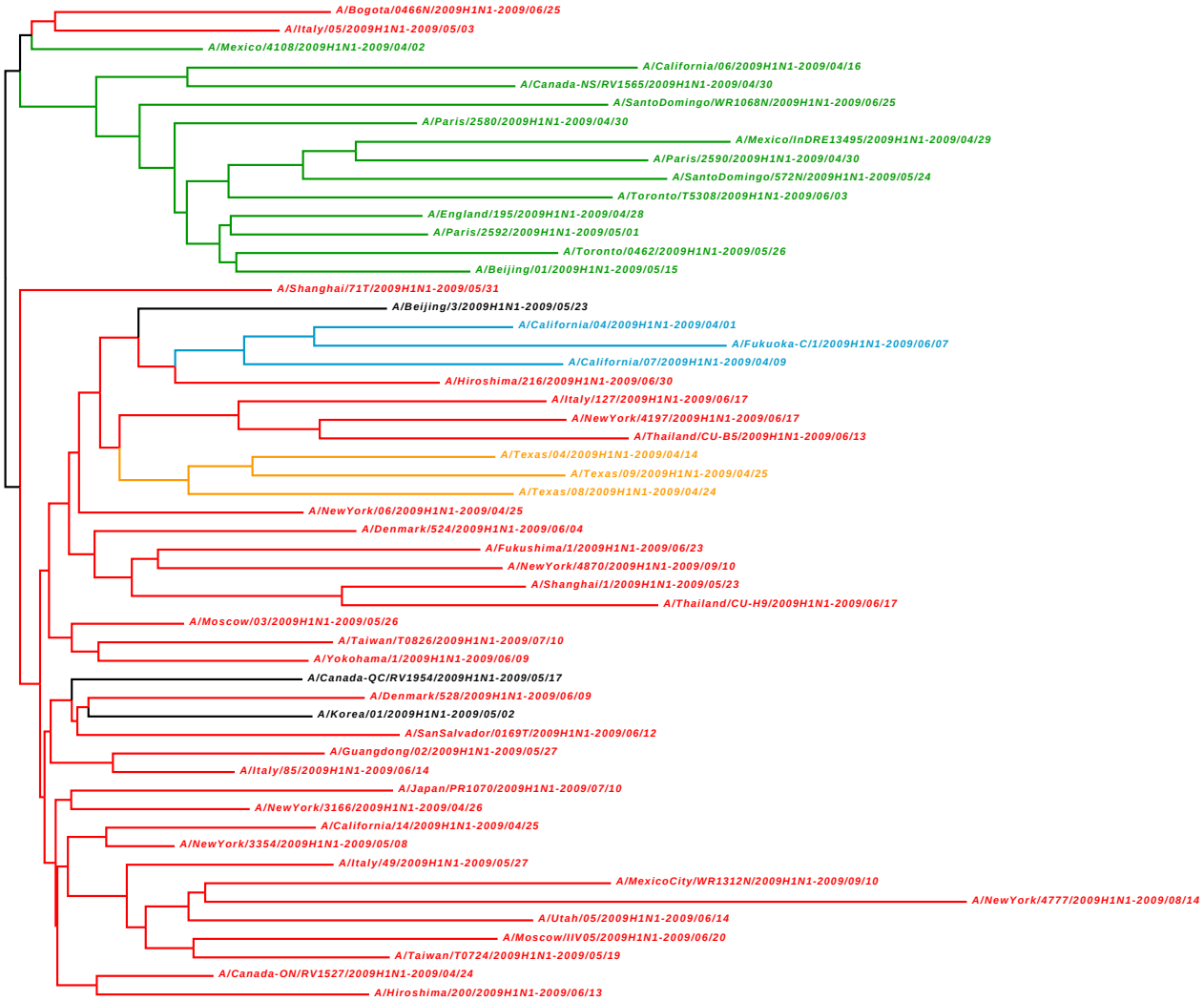


FIGURE 4.1. *Whole-genome phylogeny of the 2009 world pandemic Influenza A (H1N1) generated by USA. In green and red we point out the two main clades found in the literature [103], where the green Mexico/4108 is probably the closest isolate to the origin of the flu. In blue and orange are two of the possible early evolutions of the viral disease. In black, the organisms which in the literature do not fall into one of the two clades.*

FIGURE 4.2. *Whole-genome phylogeny of prokaryotes by USA. In red are the branches of the Archaea domain, while in green are those of the Bacteria domain. Other clusters found in the reference taxonomy are highlighted with different colors on the names of organisms (apart from the black color). Only two organisms do not fall into the correct clade: Methanosarcina acetivorans (in cyan) and Desulfovibrio vulgaris subspecies vulgaris (in black).*

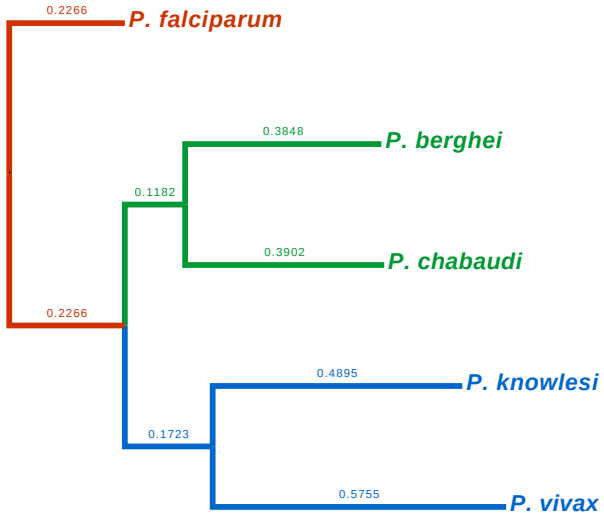


FIGURE 4.3. *Whole-genome phylogeny of the genus Plasmodium by USA, with our whole-genome distance highlighted on the branches.*

FFP for every single region of genomes. This fact has also been attested by our general method, USA: we rerun all the experiments by filtering out the subwords of length $< log_4 m$, and compared the resulting distance matrices with those previously achieved (i.e., with no filtering). In every case we observed that the matrices are almost identical. Furthermore, each unic subword has attested on average a few occurrences in each sequence, in general by attaining only one occurrence per sequence.

We can further analyze the average number of inversions and complements, where the increasing size of sequences seems to attest their values to 33 % and 19-20 %, respectively, out of the total number of unic subwords. However, this fact may be relegated to the nature of the sequences considered.

In addition, defining the number of free locations in a sequence as the number of locations that are not covered by unic subwords in a pairwise comparison, we notice that this number may vary widely with respect to the size and type of the sequences under examination. In fact, in case of sequences comparable in length, the percentage of free locations throughout each sequence ranges from 3.5 %, the average value in the case of viruses, to 15 %, e.g., for Plasmodium, while it can be higher if the sequences have different sizes. Another example is given by the 30 % of free locations obtained

Table 4.5. Main Counts for the Unic Subword Approach

Count	Influenza A	Arch. & Bact.	Plasmodium
Min genome size	12,976 bp	650 kbp	18,524 kbp
Max genome size	13,611 bp	8,350 kbp	23,730 kbp
Average genome size	13,230 bp	2,700 kbp	21,380 kbp
Irredundants $\lvert\mathcal{I}_{s_1,s_2}\rvert$	3,722	3,167 k	16,354 k
Unic subwords $\lvert\mathcal{U}_{s_1,s_2}\rvert$	60	112 k	706 k
Untied occ. in $\mathcal{U}_{s_1,s_2}$	63	138 k	761 k
Min $\lvert w\rvert$ in $\mathcal{U}_{s_1,s_2}$	6	10	12
Max $\lvert w\rvert$ in $\mathcal{U}_{s_1,s_2}$	1,615	25	266
Average $\lvert w\rvert$ in $\mathcal{U}_{s_1,s_2}$	264	14	20
Untied inversions	28 %	31 %	33 %
Untied complements	22 %	20 %	19 %

Major numbers of the USA measure, averaged over all experiments performed. The minimum and average length for the untied occurrences have been computed by removing the high-frequency irredundant common subwords present in a pairwise comparison. The percentages of inversions and complements refer to the number of untied occurrences per sequence.

in the comparison of the archaeon Pyrococcus furiosus, of size 1.9 Mbp, with Treponema pallidum subspecies pallidum, of size 1.2 Mbp, while considering as reference sequence the first organism.

Finally, we compared USA with its simpler version which does not consider inversions and complements. We rerun our experiments in the two scenarios, with and without inversions and complements. While for eukaryotes and prokaryotes we achieved comparable distance matrices and therefore identical phylogenies of their species, for viruses we achieved different phylogenies, more probably due to the nature of this latter organisms.

4.4 Discussion and Future Work

This chapter has introduced a distance-like function for the comparison of whole genomes, extending the ACS approach. We consider direct matches, inversions, and complements present in a pairwise comparison of genomes.

We have first shown that the irredundant common subwords are closely related to the Kullback-Leibler divergence estimated between two genome sequences. Moreover, we have provide an algorithm to discover the irredundant common subwords for two sequences s_1 and s_2, by exploiting a generalized suffix tree. Then, we have filtered these subwords in order to relate to each other the regions of s_1 and s_2; thus avoiding the overcount of the same parts a multiple number of times.

Preliminary experimental results have shown very good performance in the identification of major clusters for viruses, prokaryotes, and unicellular eukaryotes. In addition, one may want to implement the Block method for genomes not comparable in size, as seen in Subsection 4.1.1; this may further improve the performance of USA.

Additional experiments may be performed on multicellular eukaryotes, e.g., on genomes of mammals that are in the order of Gbp, by realizing more efficient implementations of our method, e.g., with suffix arrays. In this spirit, our advise is to exploit the simple algorithm proposed in [113] for the computation of the matching statistics L_{s_1,s_2} and L_{s_2,s_1}, where two different suffix arrays are employed; then, [1] may help us in finding the unic subwords. Since the analysis of large genomes can pose several problems in memory usage, compressed suffix arrays constructed with the Burrows-Wheeler transform can be employed to create efficient indexes [39; 76].

Finally, we can use the unic subwords to identify, for clades of genome sequences, conserved sites during evolution. This can be at the basis to answer other fundamental questions in genomics and computational molecular biology.

Chapter 5

Conclusion

In this thesis we have studied how the alignment-free notions of irredundant common motifs and of underlying motifs can solve some issues emerged in the field of computational molecular biology. We have specifically addressed three fundamental problems:

1. the classification and remote homology detection of protein sequences;

2. the comparison and filtering of degenerate motifs for simplifying the output of state-of-the-art motif discovery tools, and consequently identifying subtle biological signals in functional and conserved sites;

3. the comparison of whole genomes, in order to reconstruct the phylogeny of different species.

In Chapter 2 we have observed that almost all state-of-the-art methods for the remote protein homology detection problem are based on motifs that are not independent, and therefore the associated results are obtained using a set of redundant features, misleading the classification task. We have carefully considered these observations to develop the Irredundant Class method, based on the syntax and statistics of the irredundant common motifs with don't cares between a pair of protein sequences. We have employed our method as a kernel in support vector machines, and showed that this approach outperforms most of the previously proposed discriminative string

kernels in a well-known benchmark. In addition, we have analyzed the information properties of the irredundant common motifs, both in a theoretical framework and through empirical observations and postprocessing of motifs.

Furthermore, we have seen how the selection of possibly large shared motifs, one for each region of a set of sequences, may help us to simplify the output of one or more motif discovery tools and solve the whole-genome phylogeny reconstruction. In particular, in Chapter 3 we have studied degenerate motifs, introducing notions for the comparison and ranking of these motifs by means of the motif priority rule. This rule is a fingerprint measure based on motif length, composition, and the lexicographic order of occurrences. We have employed the motif priority rule as a filter to reduce the number and total length of degenerate motifs in output from modern discovery tools; thus enhancing their possible integration. For each region of the sequences considered we choose the motifs with the highest priority in a recursive, combinatorial approach. We call the resulting motifs, the underlying motifs. Our priority rule, together with the concept of underlying motifs, has proved to be valuable for the analysis of biological sequences, in particular of protein families, in many cases highlighting the candidate functional motifs closest to the actual ones. In addition, this approach drastically reduces the number of degenerate motifs that scientists have to manually inspect, without resorting to complex statistical methods or motif alignments.

Finally, in Chapter 4 we have assembled the concepts introduced in chapters 2 and 3, in order to enable the comparison of whole genomes. By means of the underlying-paired irredundant common subwords, also called unic subwords, we have built an efficient measure for the comparison of two entire genomes by relating to each other the regions of the two sequences. The resulting method, called Unic Subword Approach (USA), mimics and filters a measure closely related the Kullback-Leibler information divergence estimated between two genome sequences. USA has shown better performance than other state-of-the-art methods in the whole-genome phylogeny reconstruction problem, further identifying the major clusters for viruses, prokaryotes, and unicellular eukaryotes.

In summary, the scientific contribution of this thesis is given by an intense analysis of motifs that could be conserved functional sites in the biological

context. Indeed these motifs accomplish the classification and processing of massive sequences recently obtained from high-throughput technologies in molecular biology. Moreover, the above results confirm our intuition that the combination of the key biological motifs tends to be much smaller than the sequences considered, thus enabling to carry out space-conscious approaches.

Bibliography

[1] Abouelhoda, M.I., Kurtz, S., and Ohlebusch, E. (2004). Replacing suffix trees with enhanced suffix arrays. *Journal of Discrete Algorithms*, 2: 53–86.

[2] Abrahamson, K. (1987). Generalized string matching. *SIAM Journal on Computing*, 16: 1039–1051.

[3] Altschul, S.F., Madden, T.L., Schffer, A.A., Schaffer, R.A., Zhang, J., et al. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25: 3389–3402.

[4] Apostolico, A. (2002). Pattern discovery and the algorithmics of surprise. In Proceedings NATO ASI on Artificial Intelligence and Heuristic Methods for Bioinformatics, pp. 111–127.

[5] Apostolico, A. (2010). Maximal words in sequence comparisons based on subword composition. In Algorithms and Applications, T. Elomaa, H. Mannila, and P. Orponen, Eds., vol. 6060 of *Lecture Notes in Computer Science*, Springer, pp. 34–44.

[6] Apostolico, A., Bock, M.E., Lonardi, S., and Xu, X. (2000). Efficient detection of unusual words. *Journal of Computational Biology*, 7(1/2): 71–94.

[7] Apostolico, A., Comin, M., and Parida, L. (2006). Mining, compressing and classifying with extensible motifs. *Algorithms for Molecular Biology*, 1: 4.

[8] Apostolico, A., Comin, M., and Parida, L. (2010). VARUN: discovering extensible motifs under saturation constraints. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 7(4): 752–762.

[9] Apostolico, A., and Parida, L. (2004). Incremental paradigms of motif discovery. *Journal of Computational Biology*, 11: 15–25.

[10] Apostolico, A., and Tagliacollo, C. (2007). Optimal offline extraction of irredundant motif bases. In Proceedings of the Thirteenth Annual International Conference on Computing and Combinatorics (COCOON 2007), G. Lin, Ed., Springer, pp. 360–371.

[11] Auch, A., Henz, S., Holland, B., and Goker, M. (2006). Genome BLAST distance phylogenies inferred from whole plastid and whole mitochondrion genome sequences. *BMC Bioinformatics*, 7: 350.

[12] Aurrecoechea, C., Brestelli, J., Brunk, B.P., Dommer, J., Fischer, S., et al. (2009). PlasmoDB: a functional genomic database for malaria parasites. *Nucleic Acids Research*, 37(Database issue): D539–D543.

[13] Bailey, T.L. (2008). Discovering sequence motifs. In *Bioinformatics: Data, Sequence Analysis and Evolution*, J.W. Jonathan M. Keith, Ed., vol. 452 of *Methods in Molecular Biology*. Springerch. 12, pp. 231–251.

[14] Bailey, T.L., Williams, N., Misleh, C., and Li, W.W. (2006). MEME: discovering and analyzing DNA and protein sequence motifs. *Nucleic Acids Research*, 34(Web-Server-Issue): 369–373.

[15] Baisnée, P.-F., Hampson, S., and Baldi, P. (2002). Why are complementary DNA strands symmetric? *Bioinformatics*, 18(8): 1021–1033.

[16] Baldi, P., Chauvin, Y., Hunkapiller, T., and McClure, M. (1994). Hidden Markov models of biological primary sequence information. *Proceedings of the National Academy of Sciences USA*, 91: 1053–1063.

[17] Ben-Gal, I.E., Shani, A., Gohr, A., Grau, J., Arviv, S., et al. (2005). Identification of transcription factor binding sites with variable-order Bayesian networks. *Bioinformatics*, 21(11): 2657–2666.

[18] Ben-Hur, A., and Brutlag, D.L. (2003). Remote homology detection: a motif based approach. *Bioinformatics*, 19(Suppl. 1): i26–i33.

[19] Bender, M.A., and Farach-Colton, M. (2004). The level ancestor problem simplified. *Theoretical Computer Science*, 321(1): 5–12.

[20] Biswas, K.M., DeVido, D.R., and Dorsey, J.G. (2003). Evaluation of methods for measuring amino acid hydrophobicities and interactions. *Journal of Chromatography A*, 1000(1-2): 637–655.

[21] Burstein, D., Ulitsky, I., Tuller, T., and Chor, B. (2005). Information theoretic approaches to whole genome phylogenies. In Proceedings of the Ninth Annual International Conference on Research in Computational Molecular Biology (RECOMB 2005), S. Miyano, J. Mesirov, S. Kasif, et al., Eds., Springer, pp. 283–295.

[22] Chakravarty, A., Carlson, J.M., Khetani, R.S., DeZiel, C.E., and Gross, R.H. (2007). SPACER: identification of *cis*-regulatory elements with non-contiguous critical residues. *Bioinformatics*, 23(8): 1029–1031.

[23] Chor, B., Horn, D., Goldman, N., Levy, Y., and Massingham, T. (2009). Genomic DNA k-mer spectra: models and modalities. *Genome Biology*, 10(10): R108.

[24] Coatney, M., and Parthasarathy, S. (2003). MotifMiner: a general toolkit for efficiently identifying common substructures in molecules. In Proceedings of the Third IEEE Symposium on BioInformatics and BioEngineering (BIBE 2003), IEEE Computer Society, pp. 336–340.

[25] Cohen, E., and Chor, B. (to appear, 2012). Whole-genome phylogeny of multi cellular eukaryotes: A saturation of phylogenetic signal? *Proceedings of the National Academy of Sciences USA*.

[26] Cole, J.R., Chai, B., Farris, R.J., Wang, Q., Kulam, S.A., et al. (2005). The Ribosomal Database Project (RDP-II): sequences and tools for high-throughput rRNA analysis. *Nucleic Acids Research*, 33(Suppl. 1): D294–D296.

[27] Comin, M., and Verzotto, D. (2010). Classification of protein sequences by means of irredundant patterns. *BMC Bioinformatics*, 11(Suppl. 1): S16.

[28] Comin, M., and Verzotto, D. (2011). The Irredundant Class method for remote homology detection of protein sequences. *Journal of Computational Biology*, 18(12): 1819–1829.

[29] Comin, M., and Verzotto, D. (to appear, 2012). Comparing, ranking and filtering motifs with character classes: application to biological sequence analysis. In *Biological Knowledge Discovery Handbook: Preprocessing, Mining and Postprocessing of Biological Data*, M. Elloumi and A.Y. Zomaya, Eds. Wiley.

[30] Cornish-Bowden, A. (1985). Nomenclature for incompletely specified bases in nucleic acid sequences: recommendations 1984. *Nucleic Acids Research*, 13(9): 3021–3030.

[31] Crick, F. (1970). Central dogma of molecular biology. *Nature*, 227(5258): 561–563.

[32] Cristianini, N., and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

[33] Das, M., and Dai, H.-K. (2007). A survey of DNA motif finding algorithms. *BMC Bioinformatics*, 8(Suppl. 7): S21.

[34] Ding, C.H.Q., and Dubchak, I. (2001). Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics*, 17(4): 349–358.

[35] Edwards, S.V., Liu, L., and Pearl, D.K. (2007). High-resolution species trees without concatenation. *Proceedings of the National Academy of Sciences USA*, 104(14): 5936–5941.

[36] Eichhörn, J. (2007). *Applications of Kernel Machines to Structured Data*. Ph.D. Thesis, Technische Universität Berlin.

[37] Federico, M., and Pisanti, N. (2009). Suffix tree characterization of maximal motifs in biological sequences. *Theoretical Computer Science*, 410(43): 4391–4401.

[38] Felsenstein, J. (1993). *PHYLIP: phylogenetic inference package, version 3.5c*. Department of Genetics, University of Washington.

[39] Ferragina, P., and Manzini, G. (2000). Opportunistic data structures with applications. In Proceedings of the Forty-first Annual Symposium on Foundations of Computer Science (FOCS 2000), IEEE Computer Society, pp. 390–398.

[40] Fischer, M.J., and Paterson, M.S. (1974). String-matching and other products. Tech. Rep. MIT-LCS-TM-041, Massachusetts Institute of Technology.

[41] Fredriksson, K., and Grabowski, S. (2008). Efficient algorithms for pattern matching with general gaps, character classes, and transposition invariance. *Information Retrieval*, 11: 335–357.

[42] Frith, M.C., Saunders, N.F.W., Kobe, B., and Bailey, T.L. (2008). Discovering sequence motifs with arbitrary insertions and deletions. *PLoS Computational Biology*, 4(5): e1000071.

[43] Gao, Y., Yang, M., Wan, X., Mathee, K., and Narasimhan, G. (1999). Motif detection in protein sequences. In Proceedings of the Seventh Symposium on String Processing and Information Retrieval (SPIRE 1999), IEEE Computer Society, pp. 63–72.

[44] Giancarlo, R., Scaturro, D., and Utro, F. (2009). Textual data compression in computational biology: a synopsis. *Bioinformatics*, 25(13): 1575–1586.

[45] Gribskov, M., McLachlan, A.D., and Eisenberg, D. (1987). Profile analysis: detection of distantly related proteins. *Proceedings of the National Academy of Sciences USA*, 84(13): 4355–4358.

[46] Gribskov, M., and Robinson, N.L. (1996). Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching. *Computers and Chemistry*, 20: 25–33.

[47] Grossi, R., Pietracaprina, A., Pisanti, N., Pucci, G., Upfal, E., et al. (2011). MADMX: a strategy for maximal dense motif extraction. *Journal of Computational Biology*, 18(4): 535–545.

[48] Gusfield, D. (1997). *Algorithms on strings, trees, and sequences: Computer science and computational biology.* Cambridge University Press.

[49] Haussler, D. (1999). Convolution kernels on discrete structures. Tech. Rep. UCSC-CRL-99-10, University of California, Santa Cruz.

[50] Henikoff, S., and Henikoff, J.G. (1992). Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences USA*, 89(22): 10915–10919.

[51] Hou, Y., Hsu, W., Lee, M.L., and Bystroff, C. (2003). Efficient remote homology detection using local structure. *Bioinformatics*, 19: 2294–2301.

[52] Huang, J.Y., and Brutlag, D.L. (2001). The eMOTIF database. *Nucleic Acids Research*, 29: 202–204.

[53] Hulo, N., Bairoch, A., Bulliard, V., Cerutti, L., Cuche, B., et al. (2008). The 20 years of PROSITE. *Nucleic Acids Research*, 36(Database issue): D245–D249.

[54] Jaakkola, T., Diekhans, M., and Haussler, D. (1999). Using the Fisher kernel method to detect remote protein homologies. In Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology (ISMB 1999), pp. 149–158.

[55] Jensen, K.L., Styczynski, M.P., Rigoutsos, I., and Stephanopoulos, G.N. (2006). A generic motif discovery algorithm for sequential data. *Bioinformatics*, 22(1): 21–28.

[56] Jiang, C., and Pugh, B.F. (2009). Nucleosome positioning and gene regulation: advances through genomics. *Nature Reviews Genetics*, 10(3): 161–172.

[57] Jones, N.C., and Pevzner, P.A. (2004). *An Introduction to Bioinformatics Algorithms*. Computational Molecular Biology. The MIT Press.

[58] Kärkkäinen, J., Sanders, P., and Burkhardt, S. (2006). Linear work suffix array construction. *Journal of the ACM*, 53: 918–936.

[59] Karp, R.M., Miller, R.E., and Rosenberg, A.L. (1972). Rapid identification of repeated patterns in strings, trees and arrays. In Proceedings of the Fourth Annual ACM Symposium on Theory of Computing (STOC 1972), pp. 125–136.

[60] Karplus, K., Barrett, C., and Hughey, R. (1998). Hidden Markov models for detecting remote protein homologies. *Bioinformatics*, 14: 846–856.

[61] Kong, S.-G., Fan, W.-L., Chen, H.-D., Hsu, Z.-T., Zhou, N., et al. (2009). Inverse symmetry in complete genomes and whole-genome inverse duplication. *PLoS ONE*, 4(11): e7553.

[62] Kopelowitz, T., and Lewenstein, M. (2007). Dynamic weighted ancestors. In Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2007), SIAM, pp. 565–574.

[63] Krogh, A., Brown, M., Mian, I.S., Sjölander, K., and Haussler, D. (1994). Hidden Markov models in computational biology: applications to protein modeling. *Journal of Molecular Biology*, 235: 1501–1531.

[64] Kuang, R., Ie, E., Wang, K., Wang, K., Siddiqi, M., et al. (2005). Profile-based string kernels for remote homology detection and motif extraction. *Journal of Bioinformatics and Computational Biology*, 3(3): 527–550.

[65] Kulhavý, R. (1996). A Kullback-Leibler distance approach to system identification. *Annual Reviews in Control*, 20: 119–130.

[66] Kullback, S., and Leibler, R.A. (1951). On information and sufficiency. *Teh Annals of Mathematical Statistics*, 22(1): 79–86.

[67] Kurtz, S., Phillippy, A., Delcher, A.L., Smoot, M., Shumway, M., et al. (2004). Versatile and open software for comparing large genomes. *Genome Biology*, 5(2): R12.

[68] Ladunga, I., and Smith, R.F. (1997). Amino acid substitutions preserve protein folding by conserving steric and hydrophobicity properties. *Protein Engineering*, 10(3): 187–196.

[69] Lawrence, C.E., Altschul, S.F., Boguski, M.S., Liu, J.S., Neuwald, A.F., et al. (1993). Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, 262(5131): 208–214.

[70] Lempel, A., and Ziv, J. (1976). On the complexity of finite sequences. *IEEE Transactions on Information Theory*, 22(1): 75–81.

[71] Leslie, C.S., Eskin, E., Cohen, A., Weston, J., and Noble, W.S. (2004). Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4): 467–476.

[72] Leslie, C.S., Eskin, E., and Noble, W.S. (2002). The spectrum kernel: a string kernel for SVM protein classification. In Proceedings of the Seventh Pacific Symposium on Biocomputing (PSB 2002), R.B. Altman, A.K. Dunker, L. Hunter, et al., Eds., pp. 564–575.

[73] Li, M., Badger, J.H., Chen, X., Kwong, S., Kearney, P., et al. (2001). An information-based sequence distance and its ap-

plication to whole mitochondrial genome phylogeny. *Bioinformatics*, 17(2): 149–154.

[74] Liao, L., and Noble, W.S. (2003). Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships. *Journal of Computational Biology*, 10(6): 857–868.

[75] Lingner, T., and Meinicke, P. (2008). Word correlation matrices for protein sequence analysis and remote homology detection. *BMC Bioinformatics*, 9: 259.

[76] Lippert, R.A. (2005). Space-efficient whole genome comparisons with Burrows-Wheeler transforms. *Journal of Computational Biology*, 12(4): 407–415.

[77] Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., and Watkins, C.J.C.H. (2002). Text classification using string kernels. *Journal of Machine Learning Research*, 2: 419–444.

[78] Martinsen, E.S., Perkins, S.L., and Schall, J.J. (2008). A three-genome phylogeny of malaria parasites (Plasmodium and closely related genera): Evolution of life-history traits and host switches. *Molecular Phylogenetics and Evolution*, 47: 261–273.

[79] Mendes, N.D., Casimiro, A.C., Santos, P.M., Sá-Correia, I., Oliveira, A.L., et al. (2006). MUSA: a parameter free algorithm for the identification of biologically significant motifs. *Bioinformatics*, 22(24): 2996–3002.

[80] Murzin, A.G., Brenner, S.E., Hubbard, T., and Chothia, C. (1995). SCOP: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247: 536–540.

[81] Navarro, G., and Raffinot, M. (2003). Fast and simple character classes and bounded gaps pattern matching, with applications to protein searching. *Journal of Computational Biology*, 10(6): 903–923.

[82] Needleman, S.B., and Wunsch, C.D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3): 443–453.

[83] Nevill-Manning, C.G., Wu, T.D., and Brutlag, D.L. (1998). Highly specific protein sequence motifs for genome analysis. *Proceedings of the National Academy of Sciences USA*, 95: 5865–5871.

[84] Noble, W.S., and Pavlidis, P. (2002). Gist support vector machine and kernel principal components analysis software toolkit. Columbia University.

[85] Parida, L. (1998). *Algorithmic Techniques in Computational Genomics*. Ph.D. Thesis, Courant Institute of Mathematical Sciences, New York University.

[86] Parida, L. (2007). *Pattern Discovery in Bioinformatics: Theory and Algorithms*. Mathematical and Computational Biology. Chapman and Hall/CRC.

[87] Parida, L., Rigoutsos, I., Floratos, A., Platt, D., and Gao, Y. (2000). Pattern discovery on character sets and real-valued data: linear bound on irredundant motifs and an efficient polynomial time algorithm. In Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2000), SIAM, pp. 297–308.

[88] Peng, C.-H., Hsu, J.-T., Chung, Y.-S., Lin, Y.-J., Chow, W.-Y., et al. (2006). Identification of degenerate motifs using position restricted selection and hybrid ranking combination. *Nucleic Acids Research*, 34(22): 6379–6391.

[89] Pisanti, N., Crochemore, M., Grossi, R., and Sagot, M.-F. (2005). Bases of motifs for generating repeated patterns with wild cards. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2(1): 40–50.

[90] Pisanti, N., Soldano, H., Capentier, M., and Pothier, J. (2006). Implicit and explicit representation of approximated motifs. In *Algorithms for Bioinformatics*, C. Iliopoulos, K. Park, and K. Steinhofel, Eds. King's College London Press.

[91] Pisanti, N., Soldano, H., and Carpentier, M. (2005). Incremental inference of relational motifs with a degenerate alphabet. In Proceedings of the Sixteenth Annual Symposium on Combinatorial Pattern Matching (CPM 2005), pp. 229–240.

[92] Rangwala, H., and Karypis, G. (2005). Profile-based direct kernels for remote homology detection and fold recognition. *Bioinformatics*, 21(23): 4239–4247.

[93] Rigoutsos, I., and Floratos, A. (1998). Combinatorial pattern discovery in biological sequences: The TEIRESIAS algorithm. *Bioinformatics*, 14(1): 55–67.

[94] Rigoutsos, I., Floratos, A., Parida, L., Gao, Y., and Platt, D. (2000). The emergence of pattern discovery techniques in computational biology. *Metabolic Engineering*, 2: 159–177.

[95] Robinson, D.R., and Foulds, L.R. (1981). Comparison of phylogenetic trees. *Mathematical Biosciences*, 53: 131–147.

[96] Romer, K., Kayombya, G.R., and Fraenkel, E. (2007). WebMOTIFS: automated discovery, filtering and scoring of DNA sequence motifs using multiple programs and Bayesian approaches. *Nucleic Acids Research*, 35(Suppl. 2): W217–W220.

[97] Rosenthal, N., and Brown, S. (2007). The mouse ascending: perspectives for human-disease models. *Nature Cell Biology*, 9(9): 993–999.

[98] Saigo, H., Vert, J.-P., Ueda, N., and Akutsu, T. (2004). Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11): 1682–1689.

[99] Saitou, N., and Nei, M. (1987). The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4): 406–425.

[100] Shaner, M.C., Blair, I.M., and Schneider, T.D. (1993). Sequence logos: A powerful, yet simple, tool. In Proceedings of the Twenty-Sixth Annual Hawaii International Conference on System Sciences (HICSS 1993), T.N. Mudge, V. Milutinovic, and L. Hunter, Eds., vol. 1 of *Architecture and Biotechnology Computing*, IEEE Computer Society, pp. 813–821.

[101] Shawe-Taylor, J., and Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press.

[102] Shendure, J., and Ji, H. (2008). Next-generation DNA sequencing. *Nature Biotechnology*, 26(10): 1135–1145.

[103] Shiino, T., Okabe, N., Yasui, Y., Sunagawa, T., Ujike, M., et al. (2010). Molecular evolutionary analysis of the influenza A(H1N1)pdm, May–September, 2009: Temporal and spatial spreading profile of the viruses in japan. *PLoS ONE*, 5(6): e11057.

[104] Sims, G.E., Jun, S.-R., Wu, G.A., and Kim, S.-H. (2009). Whole-genome phylogeny of mammals: Evolutionary information in genic and nongenic regions. *Proceedings of the National Academy of Sciences USA*, 106(40): 17077–17082.

[105] Sims, G.E., Jun, S.-R.R., Wu, G.A., and Kim, S.-H.H. (2009). Alignment-free genome comparison with feature frequency profiles (FFP) and optimal resolutions. *Proceedings of the National Academy of Sciences USA*, 106(8): 2677–2682.

[106] Sims, G.E., and Kim, S.-H. (2011). Whole-genome phylogeny of Escherichia coli/Shigella group by feature frequency profiles (FFPs). *Proceedings of the National Academy of Sciences USA*, 108(20): 8329–8334.

[107] Sinha, S., and Tompa, M. (2002). Discovery of novel transcription factor binding sites by statistical overrepresentation. *Nucleic Acids Research*, 30(24): 5549–5560.

[108] Smith, G.J.D., Vijaykrishna, D., Bahl, J., Lycett, S.J., Worobey, M., et al. (2009). Origins and evolutionary genomics of the 2009 swine-origin H1N1 influenza A epidemic. *Nature*, 459(7250): 1122–1125.

[109] Smith, T.F., and Waterman, M.S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1): 195–197.

[110] Soldano, H., Viari, A., and Champesme, M. (1995). Searching for flexible repeated patterns using a non-transitive similarity relation. *Pattern Recognition Letters*, 16(3): 233–246.

[111] Tompa, M., Li, N., Bailey, T.L., Church, G.M., De Moor, B., et al. (2005). Assessing computational tools for the discovery of transcription factor binding sites. *Nature Biotechnology*, 23(1): 137–144.

[112] Ukkonen, E. (2009). Maximal and minimal representations of gapped and non-gapped motifs of a string. *Theoretical Computer Science*, 410(43): 4341–4349.

[113] Ulitsky, I., Burstein, D., Tuller, T., and Chor, B. (2006). The average common substring approach to phylogenomic reconstruction. *Journal of Computational Biology*, 13(2): 336–350.

[114] Vapnik, V.N. (1998). *Statistical Learning Theory*. Wiley-Interscience.

[115] Vilo, J. (2002). *Pattern Discovery from Biosequences*. Ph.D. Thesis, University of Helsinki.

[116] Vinga, S., and Almeida, J. (2003). Alignment-free sequence comparison – a review. *Bioinformatics*, 19(4): 513–523.

[117] Vishnevsky, O.V., and Kolchanov, N.A. (2005). ARGO: a web system for the detection of degenerate motifs and large-scale recognition of eukaryotic promoters. *Nucleic Acids Research*, 33(Suppl. 2): W417–W422.

[118] Wang, G., Yu, T., and Zhang, W. (2005). WordSpy: identifying transcription factor binding motifs by building a dictionary and learning a grammar. *Nucleic Acids Research*, 33(Suppl. 2): W412–W416.

[119] Waterman, M.S. (1995). *Introduction to Computational Biology: Maps, Sequences and Genomes*. Interdisciplinary Statistics. Chapman & Hall/CRC.

[120] Weiner, P. (1973). Linear pattern matching algorithms. In Proceedings of the Fourteenth IEEE Annual Symposium on Switching and Automata Theory (SWAT-FOCS 1973), IEEE Computer Society, pp. 1–11.

[121] Wijaya, E., Yiu, S.-M., Son, N.T., Kanagasabai, R., and Sung, W.-K. (2008). MotifVoter: a novel ensemble method for fine-grained integration of generic motif finders. *Bioinformatics*, 24: 2288–2295.

[122] Wu, R., Chaivorapol, C., Zheng, J., Li, H., and Liang, S. (2007). fREDUCE: detection of degenerate regulatory elements using correlation with expression. *BMC Bioinformatics*, 8(1): 399.

[123] Wu, S., and Manber, U. (1992). Fast text searching: allowing errors. *Communications of the ACM*, 35: 83–91.

[124] Wu, T.-J., Huang, Y.-H., and Li, L.-A. (2005). Optimal word sizes for dissimilarity measures and estimation of the degree of dissimilarity between DNA sequences. *Bioinformatics*, 21(22): 4125–4132.

[125] Yu, Z.-G., Zhan, X.-W., Han, G.-S., Wang, R.W., Anh, V., et al. (2010). Proper distance metrics for phylogenetic analysis using complete genomes without sequence alignment. *International Journal of Molecular Sciences*, 11(3): 1141–1154.

[126] Zhang, S., Su, W., and Yang, J. (2009). ARCS-Motif: discovering correlated motifs from unaligned biological sequences. *Bioinformatics*, 25: 183–189.

[127] Ziv, J., and Lempel, A. (1977). A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3): 337–343.