جامعة محمد بوضياف - المسيلة
Université Mohamed Boudiaf - M'sila

Reference: ……

UNIVERSITY OF MOHAMED BOUDIAF – M'SILA
FACULTY OF MATHEMATICS AND INFORMATICS
Department of Computer Science

Thesis

Submitted in partial fulfilment of the requirements for the degree of
***DOCTORATE 3rd Cycle* in Computer Science**
Option: Advanced Information Systems

By:

Bilal LOUNNAS

Subject

Discovery and extraction of motifs and/or profiles in biological sequences

Presented Publicly: 21 Jan 2016

To the jury:

D. MIHOUBI	Professor.	University Mohamed Boudiaf of M'sila	**President**
B. BOUDERAH	Professor.	University Mohamed Boudiaf of M'sila	**Reporter**
A. MOUSSAOUI	Professor.	University Farhat Abbas of Setif	**Co-Reporter**
A. BILAMI	Professor.	University of Batna	**Examiner**
M. BENMOHAMED	Professor.	University of Constantine	**Examiner**
M. BOURAHLA	Dr.	University Mohamed Boudiaf of M'sila	**Examiner**

Academic Year: 2015/2016.

Dedicated to the loving memory of my father LOUNNAS Ali.

1950 – 2006

إكتشاف وإستخراج الأنماط
في السلاسل البيولوجية

من طرف: لوناس بلال، البروفيسور بودراح إبراهيم و البروفيسور موساوي عبد الوهاب.

ملـــــخص: منذ اختراع تسلسل الحمض النووي من قبل فريدريك سانجر في النصف الثاني من سنوات السبعينيات، ازداد حجم البيانات البيولوجية منذ ذلك أضعافا مضاعفة بسبب تقدما تكنولوجيا الحاسوب، وأدى إلى وجود منطقة جديدة للبحث وهي البيوانفورماتيك (المعلومات الحيوية) .باختصار، الهدف من البيوانفورماتيك هو محاولة لوضع تصور لكيفية عمل الجزيئات و تنطبق تقنيات المعلوماتية لفهم وتنظيم المعلومات المرتبطة بهذه الجزيئات على نطاق واسع .

ولكن التحدي الأكبر هو تحليل واستخراج المعرفة من هذه المستودعات البيانات. في الواقع، قواعد البيانات هذه تشكل التراث الجيني للبشرية جمعاء، مثل الجينوم البشري وسلاسل مختلفة من النباتات وأنواع الكائنات الحية الحيوانية التي تم تحديدها حتى الآن، وهذا هو واحد من أهم التحديات في مجال البيوانفورماتيك وهو اكتشاف التوافق او التشابه في السلاسل البيولوجية من أجل تحديد وظيفة أو العائلة لهذه الجزيئات الكيميائية الحيوية (DNA، RNA، والبروتين). ومنذ ان هذا التحدي يعتمد على تحليل البيانات النصية، فخوارزميات المطابقة النصيه هي المرشح الأنسب لهذه المشكلة.

في هذه الرسالة قمنا بتصميم خوارزمية لاستخراج التشابهات في السلاسل البيولوجية من خلال فكرة جديدة تعتمد إستخدام عدة تقنيات من أجل الحصول على نتائج مرضية.

الكلمات المفتاحية: *سلاسل، التشابه، التطابق، الحمض الأمني، البروتين، التعرغ على الأنماط، إستخراج البيانات، خوارزميات سلسلة المطابقة*

Discovery and extraction of patterns

In biological sequences

By: LOUNNAS Bilal, Pr. BOUDERAH Brahim, and Pr. MOUSSAOUI Abdelouaheb.

ABSTRACT: Since the discovery of DNA sequencing by Frederick Sanger in the second half of the 70s, the volume of biological sequences since that has increased exponentially due to the advanced of computer technology, has brought to existence a new research area, bioinformatics. In short, bioinformatics attempts to conceptualize biology in terms of molecules (in the sense of physical-chemistry) and applies informatics techniques to understand and organize the information associated with these molecules on a large scale.

The biggest challenge remains to overcome, is the analysis and extraction of knowledge from these data repositories. Indeed, these databases constitute the genetic heritage of all humanity, like the human genome and different sequences of plant and animal biological species identified so far, and this is one of the most important challenges in bioinformatics is the discovery of motifs in biological sequences in order to define the function or the family of biochemical molecules (DNA, RNA, and Protein). Since this challenge depends on analyzing textual data, pattern matching algorithms are a suitable candidate to tackle this problem.

In this thesis we designed a novel motif discovery algorithm to meet the demand for finding motifs over biological sequences using pushdown automata as a mechanism of matching process alongside with a counter in an optimistic way.

Keywords: *Sequences, Motifs, Patterns, DNA, RNA, Protein, Pattern Recognition, Data Mining, Knowledge Extraction, String Matching Algorithms, Motif Discovery Algorithms.*

Découverte et Extraction de motifs et/ou profils Dans les séquences biologiques.

By: LOUNNAS Bilal, Pr. BOUDERAH Brahim, and Pr. MOUSSAOUI Abdelouaheb.

RÉSUMÉ: Depuis la découverte du séquençage de l'ADN par Frederick Sanger dans la deuxième moitié des années 70, le volume de séquences biologiques augmenté de façon exponentielle en raison de l'avancée de la technologie informatique, a apporté à l'existence d'un nouveau domaine de recherche, la bio-informatique. En bref, la bio-informatique tente de conceptualiser la biologie en termes de molécules (dans le sens de la physique-chimie) et applique des techniques informatiques de comprendre et d'organiser les informations associées à ces molécules sur une grande échelle.
La plus grande difficulté qui reste à surmonter, est l'analyse et l'extraction des connaissances à partir de ces gisements de données. En effet, ces banques de données constituent le patrimoine génétique de l'humanité entière, à l'instar du génome humain et des différentes séquences des espèces biologiques végétales et animales identifiées jusqu'alors, et ceci est l'un des défis les plus importants en bio-informatique est la découverte de motifs dans les séquences biologiques afin de définir la fonction ou la famille de molécules biochimiques (ADN, ARN, et protéine). Depuis ce défi dépend de l'analyse des données textuelles, les algorithmes de pattern matching sont un candidat approprié pour résoudre ce problème.
Dans cette thèse, nous avons conçu un algorithme de découverte de motif roman à répondre à la demande pour trouver des motifs sur des séquences biologiques en utilisant les automates à pile comme un mécanisme du processus d'appariement, à côté d'un compteur d'une manière optimiste.

Mots-clés: *Séquences, Motifs, Patterns, ADN, ARN, Protéines, la Reconnaissance des Formes, L'extraction des Connaissances, Recherche de Motifs, Extraction de Profils.*

ACKNOWLEDGEMENTS

In The Name of **ALLAH**, The Most Beneficent, The Most Merciful.

All praise belongs to **ALLAH** alone, and blessings and peace be upon the final Prophet.

At first a many thanks to my **mom** who never stopped trying to encourage me, and was very active in seeing me find my way throughout all the problems of this thesis.

Thanks to my **wife** and all my **family members** for theirs encouragements and supports during many hard times in my P.hD.

Many thanks to my supervisors **Pr. BOUDERAH Brahim** and **Pr. MOUSSAOUI Abdelouahab** for their supports, and advises.

Many and special thanks for my friends **DEBBI Hicham**, **BOUABDALLAH Abdelmoumn**, and **ARIOUA Abdallah** for thier helps and supports.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Part I

THESIS SUMMARY

This part has one chapter of general introduction, the aim of this chapter is to provide a preface of our research interest, our motivation for working on this kind of research issues, along with problems statements, contribution and thesis outline.

1

GENERAL INTRODUCTION

1.1 OVERVIEW

Pattern recognition of all types is an important and age old problem that goes back as far as 1973, and it is fundamental to many computer science disciplines. Nowadays most of the pattern recognition researche is focused on string matching due to the importance of this field in many other research fields.

Strings remain the main form of representing and exchanging data in a simpler and more efficient way in a range of areas, matching in this particular type of data is a very important process in computer science, aimed to find pieces of text called patterns P = p1p2...pm, in a given text T = t1t2...tn where n is always larger than m, and pi, ti are symbols of the alphabet. The task itself can be divided into two ways of matching, one called exact pattern matching which aims to locate the pattern P = p1p2...pm in the text T = t1t2...tn where each pi identical to each of the corresponding ti. The other way called approximate pattern matching, also called pattern matching that allows errors.

A broad range of research domains benefit from string matching techniques to solve their own problems. Biology and medicine are one of those research domains which depend frequently on the use of string matching techniques to solve problems, which can take years to be solved using basic techniques. Much of the biological information can be provided by molecular biology, such information is related to the analysis of biochemical molecules (Dna, Rna, and Protein), other information is related to thier interactions with each other (Dna-Protein, Dna-Rna...), these information help molecular biologists to solve many problems. The task of extracting this kind of information is not an easy process. However, even if we can get this informtaion, biologists cannot retrieve any useful knowledge due to the quantity and the complexities of this data. For this, the merging between computer science and computational biology happened in order to achieve the many goals of the new discipline called bioinformatics, with a global objec-

tive of handling and analyzing the biological information using computerized techniques and algorithms.

In this thesis we focus on string matching problems with exact and approximate algorithms classes and the application of this kind of algorithms on biological data as well. We proposed a novel algorithm in the context of extracting biological motif from biological sequences using string matching techniques

1.2 MOTIVATION

A number of different string matching algorithms have been developed in the last decade and much more are being developed due to the importance of these techniques in solving many computer science problems. Many of these algorithms are developed to solve one problem which is searching within text data, the only deference between them is just the runtime of each algorithm to find the researched pattern. However, there are some string matching algorithms developed to solve a particular problem such as the string matching algorithm for extracting biological patterns from biological data.

The availability of biological data is one of the most important reasons for researchers to develop and enhance string matching algorithms in the context of biological motif extraction. A massive amount of biological data has been produced due to the advancement of genome studies esspacilly in the late of 90s when the project of GenBank was been initiated, this collection of data created a workground for researchers to apply their ideas and algorithms and perform tests on real data.

Biological data can be categorized into many collections and groups, each of which have their own characteristics, for example Rna sequences are different from proteins and Dna sequences in many ways, like how an Rna sequence may have a loop sections. These differences make the development of a standard string matching algorithm to deal with all kinds of biological data an impossible task, from this perspective we have been motivated to build a string matching algorithm in the context of biological motif extraction to deal with Rna sequences that may contain loops in their structure.

1.3 RESEARCH PROBLEMS AND CONTRIBUTIONS

Even with the availability of biological data nowadays, it is a hard task to unify a model for representing this data based on categorization or based on their type. Our proposed method is targeted to work with a special kind of data, which is biological sequences that contain loops in their structure. The means of our method is to handle this types.

The problem here is that almost all biological repositories contains many kind of data without any discrimination or categorization, as a result we forced to do many preprocess on this data (such as cleaning, selection, filtering) before applied our algorithm. Another problem related to the first, we notice that some biological sequences exposed to some kind of changing (Biological sequences that have been exposed to mutation), we take this problem as a feature of our algorithm in order to deal with this in the process of extracting motif.

Before that, we also had to understand and learn basic information about the field of biology, from cells structures, component interaction, structures...ext. and sometimes we forced to learn about some advanced stuff in biology.

Our contribution was a result of the understanding of many topics within and without computer science such as pattern recognition, data mining, and computational biology. Therefore, we design and implement a new algorithm which will allow biologists to search and extract motifs from biological data using fast and powerful string matching techniques. In addition, we design the algorithm with a special finding process, of finding motifs contains loops in theirs structure, and this is very important feature for biologists.

1.4 ORGANIZATION OF THESIS

Our thesis is divided into two parts, the first one contain a litterateur topics to help the reader to understand the contribution of this thesis. The part itself contain three chapter, the first one is pattern recognition, we discussed techniques and methods for many pattern recognition problems and we focus on string matching problems due to its important for our thesis. We provide basic information about data mining in the second chapter, and for the third chapter we

talk about molecular biology, basic information of biology, and techniques and methods shared between computational biology and computer science.

The next part is where we talk about our contributions in the field of computational biology especially motif discovery algorithms, we provide a scientific paper in the fourth chapter that describe our proposed algorithms called MEPda in the context of motif discovery algorithm.

We end this thesis by a general conclusion and future works

Part II

THE LITERATURE PART

A three chapters will be provided in this part to help the reader to understand our work. The chapters are Pattern Recognition, Data Mining, and Bioinformatics. All of theme will contains definitions, state of the art, techniques, tools...ext.

2

PATTERN RECOGNITION

2.1 INTRODUCTION

Before the 1960s, pattern recognition was mostly the output of theoretical research in the area of statistics, as with everything else, the advancement of computers increased the demand for practical applications of pattern recognition [82]. In 1973 pattern recognition became a field within computer science, [97] which in turn set new demands for further applications in computer science including medical science, automatic speech recognition, classification of text into categories, the automatic recognition of handwritten postal codes on postal envelopes, automatic recognition of images of human faces, or handwriting image extraction from medical forms [96].

In a short definition, the field of pattern recognition is concerned with the automatic discovery of regularities in data through the use of computer algorithms and with the use of these regularities to take actions such as classifying the data into different categories [10].

2.2 THE PROBLEM OF PATTERN RECOGNITION

The problem of searching for patterns in data is fundamental in computer science and for so long had a successful history of development [10]. In order to concise of the description of pattern recognition problem statement we had to answer the following questions:

1. What is the problem?
2. Who has the problem?
3. What form can the resolution be in?

For the first question: what is the problem? The problem here is to find regularities (repetitions) in a set of data in order to take action such as classifying these data into different groups that share the same regularities.

Let's consider the example of recognizing handwritten digits represented on Figure 1, the problem here is to recognize this handwriting as the numerical number 15.

Figure 1: Handwrite of number 15

The question of who has the problem? Is related to pattern recognition applications, several research fields are using pattern recognition techniques to solve their own problems such as medical science where pattern recognition can be the basis for computer-aided diagnosis [96]. Another example of using pattern recognition techniques can be found in speech recognition, image analysis, bioinformatics (specifically motif discovery), and many more.

Computer science algorithms is the main form for representing the solutions for almost any pattern recognition problem. A lot of algorithms have been developed and are still being developed in order to fully solve pattern recognition problems, and because there are many different types of problems in pattern recognition a diversity of algorithms have been produced as well, each algorithm depends on each problem output.

2.3 THE PROCESS OF PATTERN RECOGNITION

Pattern recognition involves making sense or identifying the objects we see, this technique is known as the template matching hypothesis and the feature detection model.

A template is a pattern used to produce items of the same family. The template matching hypothesis suggests that input patterns are compared with templates. If there is a match, the input patterns are identified. While the Feature detection

model, suggests that the input patterns are broken down into their component parts for identification. [77]

In addition to this, the process of pattern matching depends on the type of the output, on whether learning is supervised or unsupervised. [10]

2.4 WHAT KIND OF PATTERN CAN BE RECOGNIZED

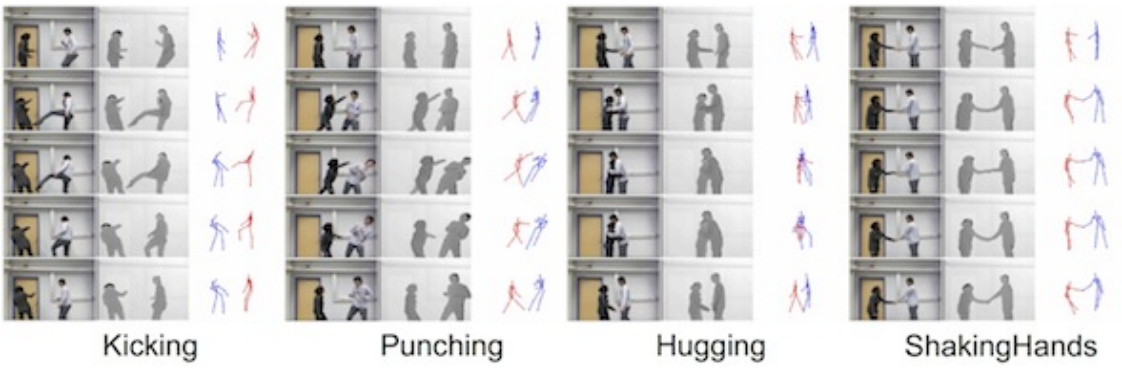
Due to the demand, information handling and retrieval are becoming increasingly important, and that has pushed Pattern recognition to the high edge of today's engineering applications and research. This evolution has made Pattern recognition an integral part of most machine intelligence systems. Therefore there are many kinds of patterns that can be recognized using the appropriate techniques.

2.4.1 *Shape matching*

The definition of shape matching can be very broad, mathematicians typically define shape matching as an equivalent class under a group that shares the same similarities. This definition can only tell us if the shape-one is the same as shape-two or not [81]. For a computer science problem we need more than that, an extensive survey of shape matching in computer vision can be found in [74] [42], in a short version the survey had divided shape matching into two groups, the first one is feature-based, which involves the use of spatial arrangements of extracted features, and the second one brightness-based, which makes more direct use of pixel brightnesses.

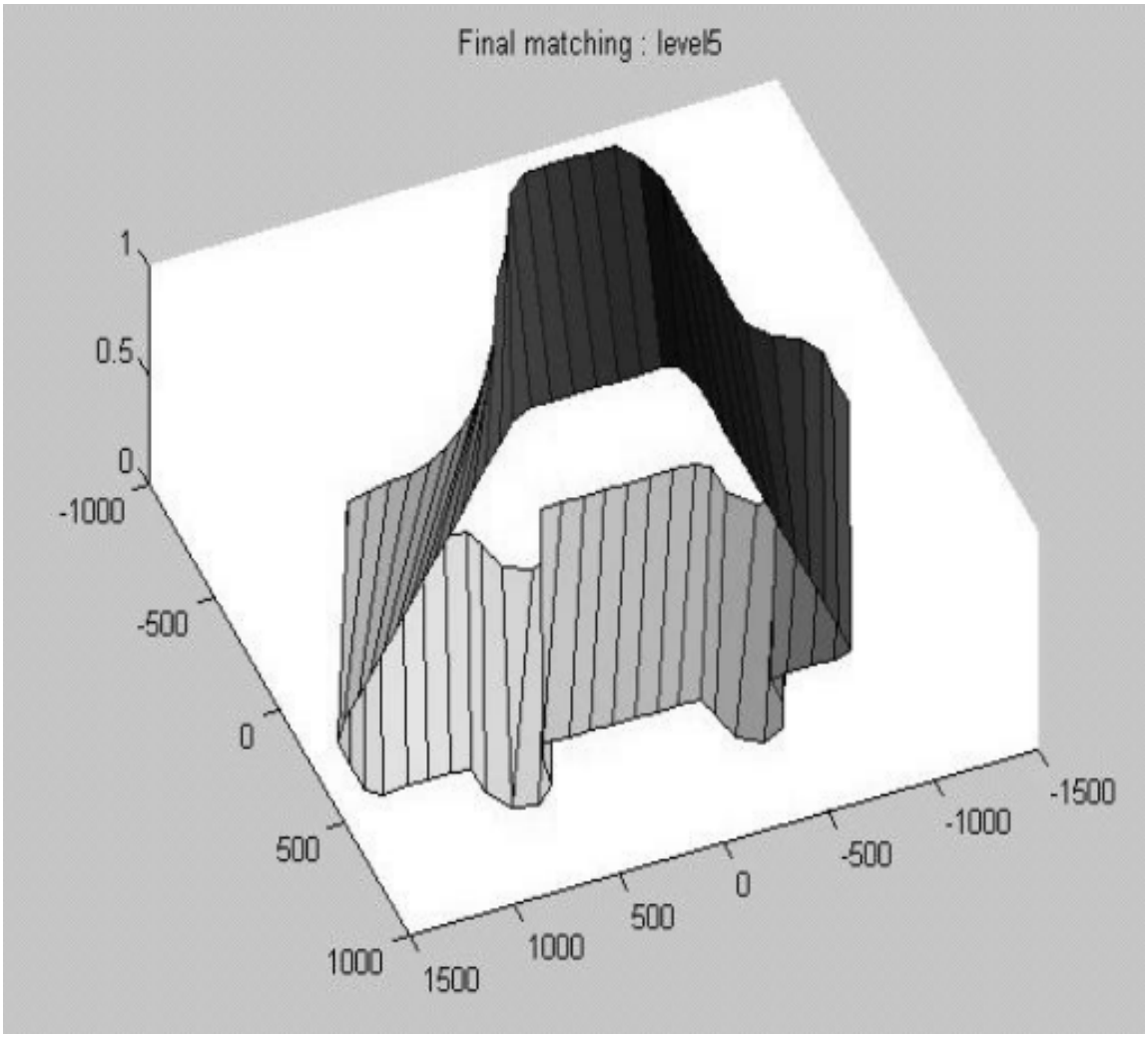
Human activity recognition, and object recognition are very important examples of shape recognition. Figure 2 illustrates two-person Interaction Detection using Body-Pose Features and Multiple Instance Learning [52].

Figure 2: Human activity recognition



[Patrice and Mounier] uses Wavelet Multi-Resolution Analysis as a pattern recognition technique to detect shapes such as cars, Figure 3.

Figure 3: Car recognition

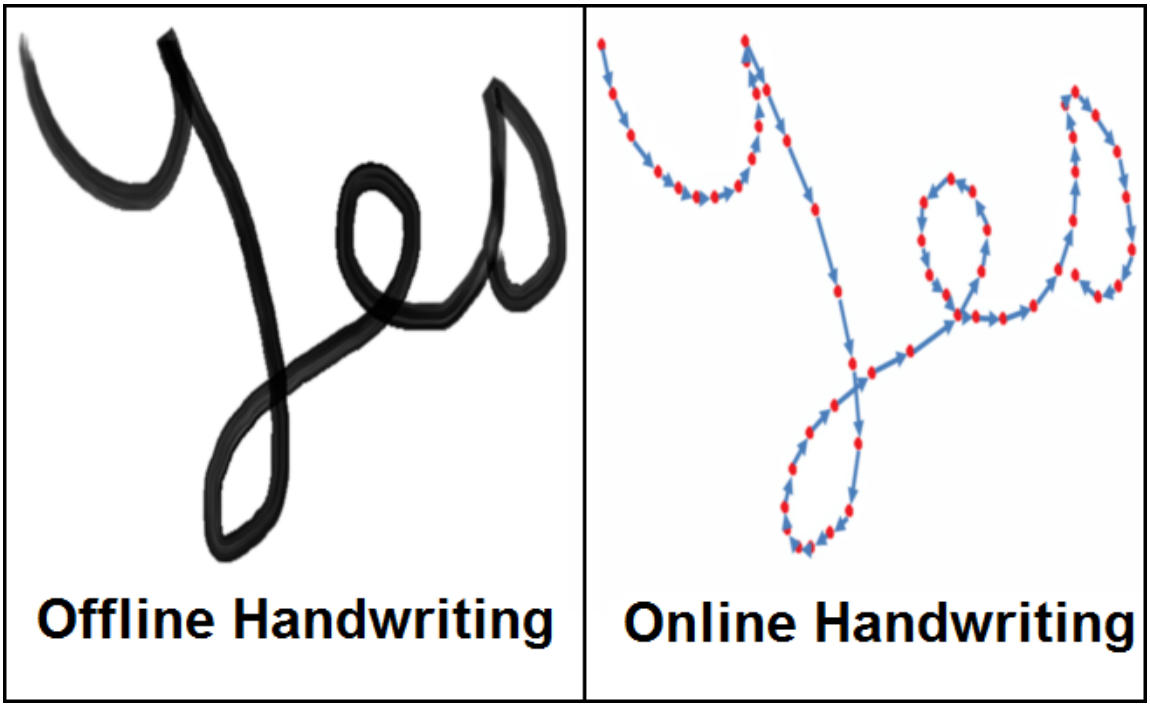


2.4.2 *Handwriting matching*

The technique by which a computer system can recognize characters and other symbols written by hand [93]. Although, the uniqueness of each person's handwriting has long been observed, techniques and methods have been developed for handwriting matching over many decades [79]

Handwriting recognition methods can generally be classified into offline and online recognition. While offline handwriting recognition deals with a bitmap presentation of the handwriting, online handwriting recognition uses the pen trajectory as input [44] [2].

Figure 4: Online and offline handwriting



2.4.3 *String matching*

Before we take deep steps into string matching, let's begin with some basic elements in this field.

2.4.3.1 *Text*

Text is a finite series of symbols from a given alphabet, where the alphabet is a nonempty finite set whose elements are called symbols. In computer science, text remains the main form of representing and exchanging data in a simpler and more efficient way in a range of areas [63] [35]. Text can be defined as a human-readable sequence of characters and is usually distinguished from non-character encoded data, such as graphics or images [76].

2.4.3.2 *String matching definition*

String matching aimed to find a piece of text called pattern P = p1p2...pm, in a given text T = t1t2...tn where n is always larger than m, and pi, ti are symbols of the alphabet [63].

2.4.3.3 *String matching classification*

There are many classification approaches of string matching algorithms in the literatures, this diversities are caused by choosing the comparison factors that discriminate the algorithms. Some classifications are dependent on the types of data representations other than the types of output results...ext.

However, the most known and used classification divides string matching algorithms into two group, word-based algorithms and approximate algorithms.

A very good survey in that context of string matching algorithms classification has been proposed in the literatures by [35].

1. Exact string matching: Also known as word-based matching algorithms, exact string matching find all the occurrences of a pattern p in a given text and reject any other words that have one or more characters mismatch compared to the pattern. The most known classification of exact string matching algorithms is classical methods, deterministic finite automata, bit-parallelism and hashing string matching algorithms.

   Classical methods are based on character comparison, deterministic finite automata methods uses automata, a data structure that stores all the suffixes or prefixes of a string, this enables fast string matching, Bit parallelism uses the essential parallelism of the bit manipulations inside com-

puter words to perform many operations in parallel, and finally hashing methods provide a simple method to avoid a quadratic number of character comparisons in most practical situations

Exact matching algorithm can run in O (n + m) where n the length of the input string and m is the length of a given pattern.

Table 1 illustrates some exact string matching algorithms with different classification

Algorithm	Classification	Reference
Wu-Manber for Single Pattern Matching	Classical	[87]
Boyer-Moore-Horspool with q-grams	Classical	[50]
Genomic Rapid Algo for String Pm	Classical	[25]
SSEF	Classical	[55]
Wide Window	Automata based	[58]
Linear DAWG Matching	Automata based	[58]
Improved Linear DAWG Matching	Automata based	[21]
Simplified Forward Backward Oracle Matching	Automata based	[33]
Forward SBNDM	Bit-Parallel	[34]
Small Alphabet Bit-Parallel	Bit-Parallel	[100]
Bit-Parallel Wide-Window2	Bit-Parallel	[15]
Factorized BNDM	Bit-Parallel	[14]
Tailed-Substring	Hashing	[13]

Table 1: Exact string matching algorithms

2. Approximate algorithms: Also known as string matching algorithms that allow errors, and it is a generalization of a exact string matching, the only difference between the two is that approximate string matching tries to find all substrings of a text string close to a given pattern string.

   Many classification approaches are proposed in the literatures for classifying the approximate string matching, for example there are classifications that are based on data representation of the pattern.

   Other classification approaches of approximate string matching algorithms can be based on the different methods employed in the searching phase, this classification divides the approximate string matching into five groups: Classical methods, counting methods, Deterministic Finite Automata Methods, Bit-Parallelism Methods, and Filtering methods.

   Approximate matching algorithm can run in O (nm) or O (nm/w), where r is the error threshold and w is the size of a computer word.

2.5 PATTERN RECOGNITION APPLICATIONS

The following five applications are only a sample from a much larger number of possible pattern recognition applications.

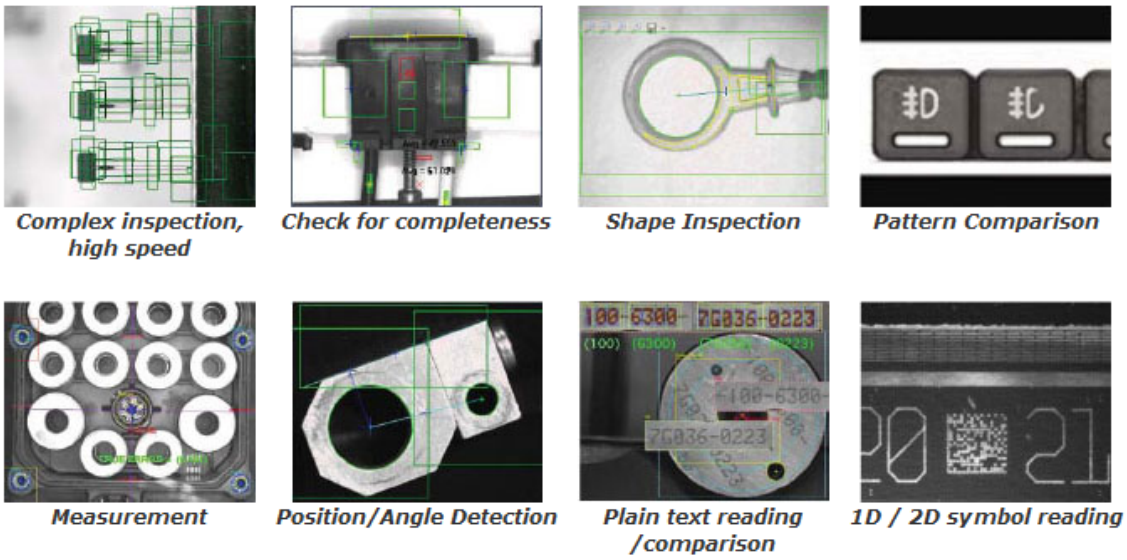
2.5.1 *Machine vision*

The definition of machine vision can be very complicated because it is concerned with a very diverse range of technologies and applications [39]

In short machine vision is the ability of a computer to produces description based on an input feed such as camera.

An example of machine vision is the systems that inspect silicon wafers, processor chips, and sub-components such as resistors and capacitors at high speeds with precision and accuracy based on pattern recognition techniques. Figure 5

Figure 5: Machine vision application based on pattern recognition techniques

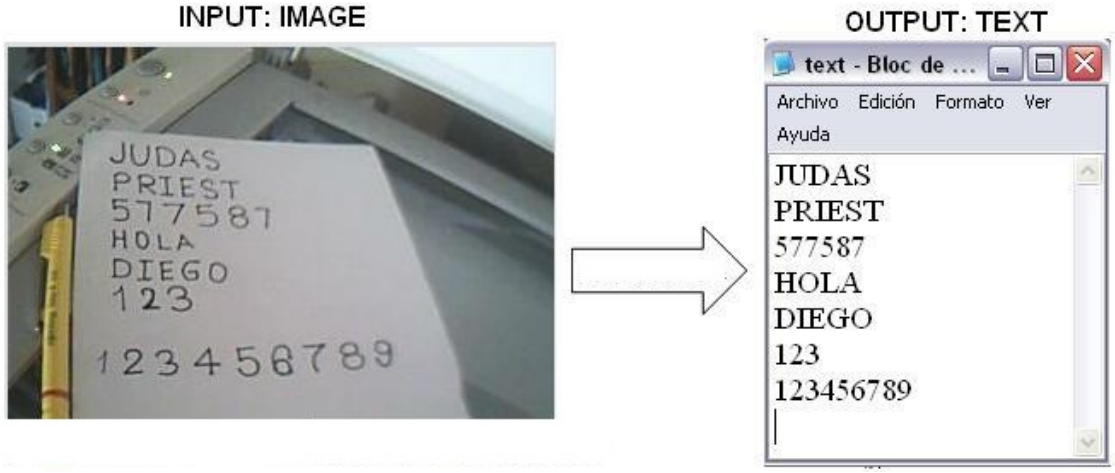


2.5.2 *Character recognition OCR*

OCR stands for Optical Character Recognition, and it is the electronic conversion of a typewritten or a printed text into encoded text.

The process of OCR is divided into two separate parts, the first one involves a hardware component in order to scan typewritten paper, and save it as a scanned image. the second part involves pattern recognition techniques in order to convert the scanned image into encoded text that can be dealt later with some kind of text-editing software.

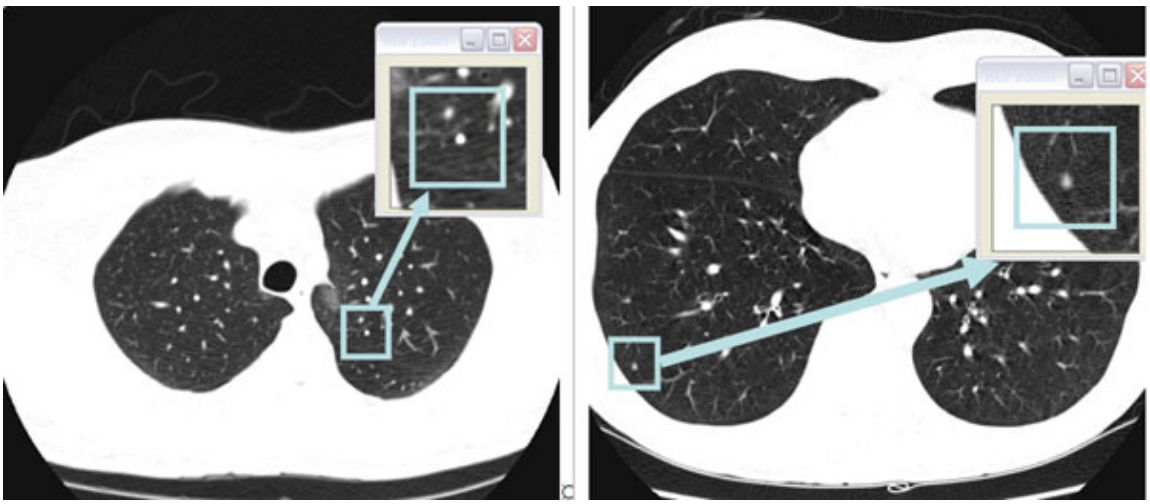
Figure 6: OCR example



2.5.3 *Computer-aided diagnosis*

CAD is one of the important applications of pattern recognition, aiming at assisting doctors in making diagnostic decisions through analysing medical data such as X-rays,computed tomographic images, ultrasound images, electrocardiograms (ECGs), and electroencephalograms (EEGs).

Figure 7: The figure shows two CAD marks on thoracic CT scans: The CAD mark on the left detects a difficult lung nodule that was missed by all radiologists without CAD in a reader study involving six experienced radiologists. The CAD mark on the right detects a relatively easy lung nodule that was detected by all six readers without CAD.



2.5.4 *Speech recognition*

A great deal of research and development has been invested in the context of speech recognition.

Scientists, researchers and engineers looking to build intelligent machines that understand spoken information, due to speech being the most natural way of human communication and information exchange.

Figure 8: Speech recognition in cars

As we mentioned earlier those are only a few examples of a larger number of possible applications in the domain of pattern recognition. Typically, we refer to other applications like fingerprint identification, signature authentication, and face and gesture recognition.

3

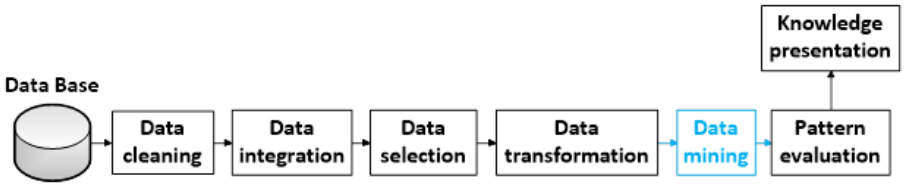
DATA MINING

3.1 WHAT IS DATA MINING?

DM is the process of discovering interesting knowledge from large amounts of data stored in databases, data warehouses, or other information repositories, and summarizing it into useful information that can be used to predict the future.

In other view data mining as simply an essential step in the process of knowledge discovery. Knowledge discovery as a process is depicted in Figure 9 and consists of an iterative sequence of the following steps [47]:

Figure 9: Knowledge discovering process



1. Data cleaning: to remove noise and inconsistent data;
2. Data integration: where multiple data sources may be combined;
3. Data selection: where data relevant to the analysis task are retrieved from the database;
4. Data transformation: where data are transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations, for instance;
5. Data mining: an essential process where intelligent methods are applied in order to extract data patterns;
6. Pattern evaluation: to identify the truly interesting patterns representing knowledge based on some interestingness measures;

7. Knowledge presentation: where visualization and knowledge representation techniques are used to present the mined knowledge to the user.

3.2 WHY DATA MINING IS IMPORTANT?

The steady and amazing progress of computer hardware technology has led to large supplies of powerful and affordable computers, data collection equipment, and storage media. This technology provides a great boost to the database and information industry, and makes a huge number of databases and information repositories available for transaction management, information retrieval, and data analysis. [47]

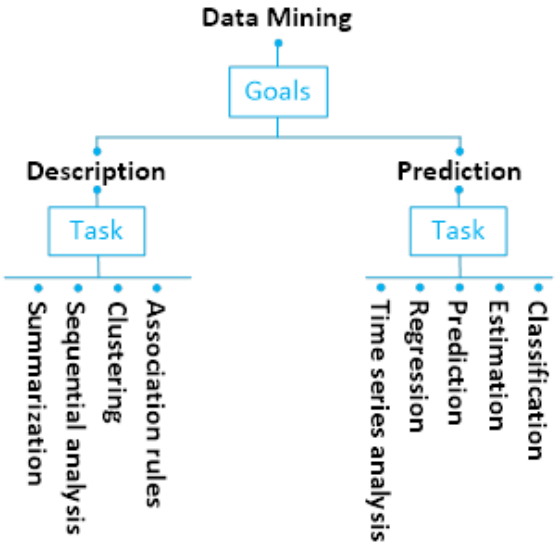
This evolution in data repositories makes it easier to collect millions, billions and even trillions pieces of information, this huge amount of data makes it a difficult task to extract useful information for particular database user management.

Data mining solves this problem by locking up the wanted data and not the complete data in the database, and extracting useful information using different techniques.

3.3 DATA MINING TASKS

Data mining can be categorized into tasks, according to different goals of a data mining practitioner. The two "high-level" primary goals of data mining, in practice, are prediction and description. [47] [48] [68] [36] [27]

Figure 10: Data mining task and goals [29]



3.3.1 *Classification*

Classification is one of the most common data mining tasks. The world is always classifying, categorizing and grading in order to understand and communicate. Classification is the activity of examining the features of a presented object and assigning it to one of a predefined set of classes.

3.3.2 *Estimation*

Estimation is often used to perform a classification task. Classification deals with different types of input data. Estimation deals with continuous values. Given some input data, estimation will give a value for some unknown continuous variable such as credit card balance.

3.3.3 *Prediction*

Prediction is the same as classification or estimation. It only differs in that the records are classified according to some predicted future behavior or estimated future value.

In other way the prediction task can predict the unknown value of one variable, based on known values of other variables.

3.3.4 *Time series analysis*

A time series analysis works on specific data that consists of sequences of values or events obtained over repeated measurements of time, to find correlation relationships within time-series data, and to extract similar or regular patterns for prediction purposes.

3.3.5 *Association rules*

The task of association rules is to find all combinations of items that are found in a sufficient number of examples, and filter them by specifying the minimum support (how often a combination occurred overall) and the minimum confidence (how often the association rule holds true in the data set) respectively.

The data set for the association rules task is usually a set of transactions, where each transaction is a set of objects.

3.3.6 *Clustering*

Clustering refers to the grouping of records, observations, or cases into classes, each one of theme contains objects that share the same properties.

The difference between clustering and classification is that clustering does not depend on predefined classes. In classification, each record is assigned to a predefined class. In clustering, there are no predefined classes and no examples. The records are grouped together based on self-similarity.

3.3.7 *Summarization*

Summarization techniques can be used to identify the typical properties of your data and highlight which data values should be treated as noise or outliers.

3.4 DATA MINING ALGORITHMS

Data Mining is concerned with the development of algorithms in order to achieve the objectives of each data mining task, the following table Table 2 shows a set of algorithms that have been widely used in the data mining community. [98]

Algorithm	Task
C4.0	Classification
K-Means	Clustering
SVM	Classification and regression
Apriori	Association rules
EM	Estimation
PageRank	Classification
AdaBoost	Classification and regression
kNN	Clustering
NaÃ¯ve Bayes	Estimation
CART	Classification

Table 2: Data mining's most known algorithms and their classification

Many algorithms have been developed in the context of data mining, and many other are still under development due to the sensitivity and wideness of data mining applications.

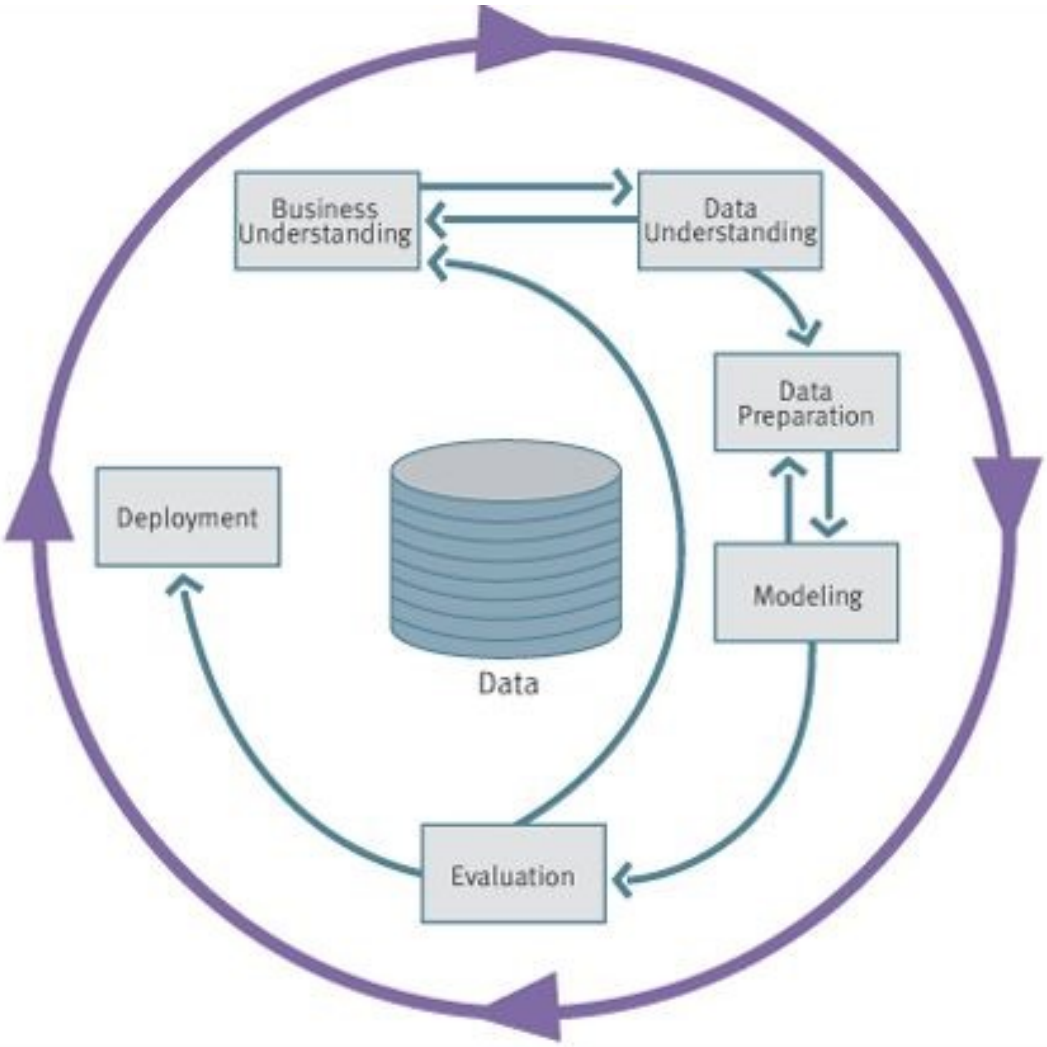
3.5 THE DATA MINING PROCESS

In the early 90's, the growing interest in data mining brought up to light the absence of a "methodology" for knowledge discovery that would be neutral to any application and tool. This is how the CRISP-DM (Cross Industry Standard Process for Data Mining) initiative was born.

Starting from the knowledge discovery processes used in early data mining projects, CRISP-DM defined and validated a data mining process that could be applicable in any industry sector. This methodology should make large data min-

ing projects faster, cheaper, more reliable and more manageable. However, even small scale data mining investigations can benefit from using it. [17]

Figure 11: Phases of the CRISP-DM reference model



The current process model for data mining provides an overview of the life cycle of a data mining project. It contains the phases of a project, their respective tasks and relationships between these tasks. At this description level, it is not possible to identify all relationships. Essentially, relationships could exist between any data mining task depending on the goals, the background and interest of the user and most importantly on the data.

The life cycle of a data mining project consists of six phases. Figure 11 shows the phases of a data mining process. The sequence of the phases is not rigid. Mov-

ing back and forth between the different phases is always required. It depends on the outcome of each phase which phase or which particular task of a phase has to be performed next. The arrows indicate the most important and frequent dependencies between phases.

3.5.1 *Business understanding*

This initial phase focuses on understanding the project objectives and requirements from a business perspective, then converting this knowledge into a data mining problem definition and a preliminary plan designed to achieve the objectives.

3.5.2 *Data understanding*

The data understanding phase starts with an initial data collection and proceeds with activities in order to get familiar with the data, to identify data quality problems, to discover first insights into the data or to detect interesting subsets to form hypotheses for hidden information.

3.5.3 *Data preparation*

The data preparation phase covers all activities to construct the final dataset (data that will be fed into the modeling tool(s)) from the initial raw data. Data preparation tasks are likely to be performed multiple times and not in any prescribed order. Tasks include table, record and attribute selection as well as transformation and cleaning of data for the modeling tools.

3.5.4 *Modeling*

In this phase, various modeling techniques are selected and applied, and their parameters are calibrated to optimal values. Typically, there are several techniques for the same data mining problem type. Some techniques have specific require-

ments on the form of data. Therefore, stepping back to the data preparation phase is often necessary.

3.5.5 *Evaluation*

At this stage in the project you have built a model (or models) that appears to have high quality from a data analysis perspective. Before proceeding to final deployment of the model, it is important to more thoroughly evaluate the model and review the steps executed to construct the model to be certain it properly achieves the business objectives. A key objective is to determine if there is some important business issue that has not been sufficiently considered. At the end of this phase, a decision on the use of the data mining results should be reached.

3.5.6 *Deployment*

The creation of the model is generally not the end of the project. Even if the purpose of the model is to increase knowledge of the data, the knowledge gained will need to be organized and presented in a way that the customer can use it. It often involves applying "live" models within an organization's decision making processes, for example in real-time personalization of Web pages or repeated scoring of marketing databases. However, depending on the requirements, the deployment phase can be as simple as generating a report or as complex as implementing a repeatable data mining process across the enterprise. In many cases it is the customer, not the data analyst, who carries out the deployment steps. However, even if the analyst will not carry out the deployment effort it is important for the customer to understand up front what actions need to be carried out in order to actually make use of the created models.

3.6 ARCHITECTURE OF DATA MINING SYSTEMS

We agree that data mining is a step in the knowledge discovery process. However, in industry, in media, and in the database research milieu, the term data mining

is becoming more popular than the wider term of knowledge discovery from data. [47]

Based on this view, the architecture of a typical data mining system may have the following major components Figure 12:

3.6.1 *Database, data warehouse, World Wide Web, or other information repositories*

This is one or a set of databases, data warehouses, spreadsheets, or other kinds of information repositories. Data cleaning and data integration techniques may be performed on the data.

3.6.2 *Database or data warehouse server*

The database or data warehouse server is responsible for fetching the relevant data, based on the user's data mining request.

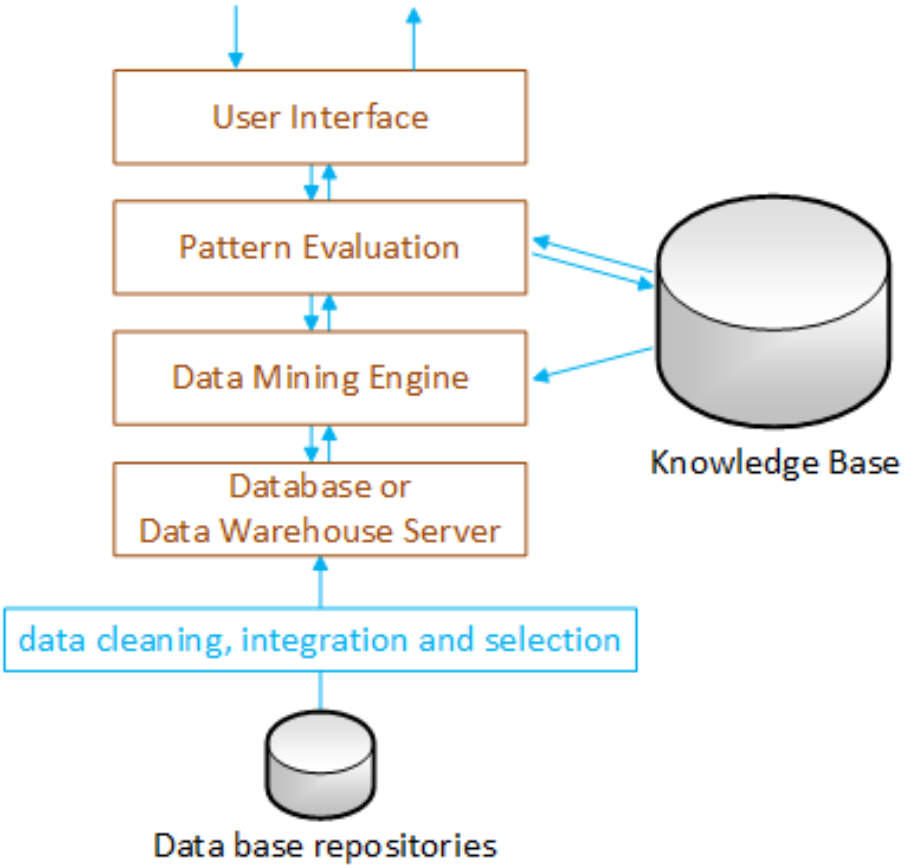
3.6.3 *Knowledge base*

This is the domain knowledge that is used to guide the search or evaluate the interestingness of resulting patterns. Such knowledge can include concept hierarchies, used to organize attributes or attribute values into different levels of abstraction. Knowledge such as user beliefs, which can be used to assess a pattern's interestingness based on its unexpectedness, may also be included.

3.6.4 *Data mining engine*

This is essential to the data mining system and ideally consists of a set of functional modules for tasks such as characterization, association and correlation analysis, classification, prediction, cluster analysis, outlier analysis, and evolution analysis.

Figure 12: Architecture of a typical data mining system [47]



3.6.5 *Pattern evaluation module*

This component typically employs interestingness measures and interacts with the data mining modules so as to focus the search toward interesting patterns. It may use interestingness thresholds to filter out discovered patterns. Alternatively, the pattern evaluation module may be integrated with the mining module, depending on the implementation of the data mining method used. For efficient data mining, it is highly recommended to push the evaluation of pattern interestingness as deep as possible into the mining process so as to confine the search to only the interesting patterns.

3.6.6 *User interface*

This module communicates between users and the data mining system, allowing the user to interact with the system by specifying a data mining query or task, providing information to help focus the search, and performing exploratory data mining based on the intermediate data mining results. In addition, this component allows the user to browse database and data warehouse schemas or data structures, evaluate mined patterns, and visualize the patterns in different forms.

3.7 THE DIFFERENT KINDS OF DATA MINING

The growth of data in various complex forms (e.g., semi-structured and unstructured, spatial and temporal, hypertext and multimedia) has been explosive, owing to the rapid progress of data collection and advanced database system technologies and the World Wide Web. [47]

The purpose of data mining is to extract knowledge from operational data base systems like data warehouses, but as the kinds of data has been changed into complex forms, mining such complex types of data became an increasingly important task in data mining.

3.7.1 *Graph mining*

As a general data structure, graphs have become increasingly important in modeling sophisticated structures and their interactions, such as circuits, chemical compounds, protein structures, biological networks, social networks and workflows. And with the increasing demand on the analysis of these kinds of structured data, graph mining has become an active and important theme in data mining.

3.7.2 *Spatial Data Mining*

A spatial database stores a large amount of space-related data, such as maps, preprocessed remote sensing or medical imaging data, and VLSI chip layout

data. Spatial databases have many features distinguishing them from relational databases.

Spatial data mining refers to the extraction of knowledge, spatial relationships, or other interesting patterns not explicitly stored in spatial databases.

3.7.3 *Multimedia Data Mining*

A multimedia database system stores and manages a large collection of multimedia data, such as audio, video, image, graphics, speech. The mining of multimedia data is to extract patterns on these kinds of data.

3.7.4 *Text Mining*

Most of the previous studies of data mining have focused on structured data, such as relational, transactional, and data warehouse data. However, in reality, a substantial portion of the available information is stored in text databases (or document databases), which consist of large collections of documents from various sources, such as news articles, research papers, books, digital libraries, e-mail messages, and Web pages.

Text mining tasks can be performed on the extracted keywords, tags, or semantic information. These include document clustering, classification, information extraction, and association analysis.

3.7.5 *Mining the World Wide Web*

The World Wide Web serves are a huge, widely distributed, global information service centers for news, advertisements, consumer information, financial management, education, government, e-commerce, and many other information services. The Web also contains a rich and dynamic collection of hyperlink information and Web page access and usage information, providing rich sources for data mining.

Web mining is a more challenging task that searches for Web structures, ranks the importance of Web contents, discovers the regularity and dynamics of Web

contents, and mines Web access patterns. However, Web mining can be used to substantially enhance the power of a Web search engine, classify Web documents, and resolve many ambiguities and subtleties raised in keyword-based Web search.

3.8 DATA MINING APPLICATIONS

As a young research field, data mining has made broad and significant progress since its early beginnings in the 1980s. Today, new and increasingly innovative applications for it are emerging. [47]

3.8.1 *Data Mining for Finance*

Most banks and financial institutions offer a wide variety of banking services (such as checking and savings accounts for business or individual customers), credit (such as business, mortgage, and automobile loans), and investment services (such as mutual funds). Some also offer insurance services and stock investment services.

Financial data collected in the banking and financial industry are often relatively complete, reliable, and of high quality, which facilitates systematic data analysis and data mining.

3.8.2 *Data mining for the Retail Industry*

The retail industry is a major application area for data mining, since it collects huge amounts of data on sales, customer shopping history, goods transportation, consumption, and service. The quantity of data collected continues to expand rapidly, especially due to the increasing ease, availability, and popularity of business conducted on the Web, or e-commerce. Retail data mining can help identify customer buying behaviors, discover customer shopping patterns and trends, improve the quality of customer service, achieve better customer retention and satisfaction, enhance goods consumption ratios, design more effective goods transportation and distribution policies, and reduce the cost of business.

3.8.3 *Data Mining for the Telecommunication Industry*

The telecommunication industry has quickly evolved from offering local and long-distance telephone services to providing many other comprehensive communication services, including fax, pager, cellular phone, Internet messenger, images, e-mail, computer and Web data transmission, and other data traffic. The integration of telecommunication, computer network, Internet, and numerous other means of communication and computing is also underway. Moreover, with the deregulation of the telecommunication industry in many countries and the development of new computer and communication technologies, the telecommunication market is rapidly expanding and is highly competitive. This creates a great demand for data mining in order to help understand the business involved, identify telecommunication patterns, catch fraudulent activities, make better use of resources, and improve the quality of the service.

3.8.4 *Data Mining for Biology*

The past decade has seen an explosive growth in genomics, proteomics, functional genomics, and biomedical research. Examples range from the identification and comparative analysis of the genomes of humans and other species (by discovering sequencing patterns, gene functions, and evolution paths) to the investigation of genetic networks and protein pathways, and the development of new pharmaceuticals and advances in cancer therapies. Biological data mining has become an essential part of a new research field called bioinformatics.

3.8.5 *Data mining for Intrusion Detection*

The security of our computer systems and data is at continual risk. The extensive growth of the Internet and increasing availability of tools and tricks for intruding and attacking networks have prompted intrusion detection to become a critical component of network administration. Most commercial intrusion detection systems are limiting and do not provide a complete solution. The current traditional

intrusion detection systems face many limitations. This has led to an increased interest in data mining for intrusion detection.

3.8.6 *Data mining for Education*

Known as Educational Data Mining, or EDM, is an emerging discipline, concerned with developing methods for exploring the unique types of data that come from educational settings, and using those methods to better understand students, and the settings which they learn in, in order to improve education. [95]

3.9 DATA MINING TOOLS

Data mining is a relatively young field with many issues that still need to be researched in depth, many off-the-shelf data mining system products and domain-specific data mining application softwares are available.

The question is "What kind of system should I choose?" Some people think that all data mining system share the same well-defined operations and a standard query language, or the same functionality, but unfortunately, this is far from reality. Many data mining system have little in common, the majority work completely differently.

The following data mining system provides multiple data mining functions and explores multiple knowledge discovery techniques. Some of them are commercial and some are free "open source". [47]

3.9.1 *SAS Enterprise Miner*

SAS Enterprise Miner streamlines the data mining process to create highly accurate predictive and descriptive models based on analysis of vast amounts of data from across the enterprise. It offers a rich, easy-to-use set of integrated capabilities for creating and sharing insights that can be used to drive better decisions.

3.9.2 *Intelligent Miner*

Is an IBM data mining product that provides a wide range of data mining functions, including association mining, classification, regression, predictive modeling, deviation detection, clustering, and sequential pattern analysis. It also provides an application toolkit containing neural network algorithms, statistical methods, data preparation tools, and data visualization tools. Distinctive features of Intelligent Miner include the scalability of its mining algorithms and its tight integration with IBM's DB2 relational database system.

3.9.3 *Clementine, from SPSS*

Has become IBM SPSS Modeler. Provides an integrated data mining development environment for end users and developers. Multiple data mining functions are incorporated into the system.

3.9.4 *Statistica Data Miner from Statsoft*

The STATISTICA Data Analysis and Data Mining Platform, including the STATISTICA Data Miner software, offers the most comprehensive and effective system of user-friendly tools for the entire data mining process - from querying databases to generating final reports. StatSoft's data mining and predictive modeling software is available in single workstation, multiple-user (concurrent user licensing), and Enterprise editions. [1]

3.9.5 *Oracle Data Mining (ODM)*

An option to Oracle Database 10g Enterprise Edition, provides several data mining functions, including association mining, classification, prediction, regression, clustering, and sequence similarity search and analysis. Oracle Database 10g also provides an embedded data warehousing infrastructure for multidimensional data analysis.

3.9.6 *Microsoft SQL Server 2008R2*

Mining historical data can provide new insights and form a reliable basis for accurate forecasting. Use predictive analysis in Microsoft SQL Server R2 2008 to inform your decisions.

3.9.7 *Weka*

Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes.

3.9.8 *RapidMiner*

Is an environment for data mining, text mining, predictive analytics, and business analytics. It is used for research, education, training, rapid prototyping, application development, and industrial applications.

It is distributed under the AGPL open source license and has been hosted by SourceForge since 2004. [94]

Many other commercial and open source data mining tools has been developed, some of theme for research other for commercial purpose.

4

BIOINFORMATICS

4.1 INTRODUCTION

A plenty of biological tools are indispensable in modern biology used by biologists gathering biological data every day to help synthesize it and integrate various types of information in the process of answering a particular biological question. In the last decade the field of biology witnessed a data explosion, due to the huge data recording by biologists using some of those tools.

The evolution of biological data in terms of size or nature has caused an appearance of a new concept on the research area: Bioinformatics, which will be more clearly defined below, is the discipline of quantitative analysis of information relating to biological macromolecules with the aid of computers [99].

The major reason why bioinformatics has become a very active and important research field was the advancement of genome studies that produced amounts of biological data, this explosion of genome sequence information has generated a demand for efficient tools, algorithms and database systems to analyse it.

4.2 WHAT IS BIOINFORMATICS?

The evolution of biological data in terms of size and nature, caused by the recording of huge amounts of data by biologists using many different techniques [31], has caused an appearance of a new concept on the research area: bioinformatics. In short, bioinformatics is conceptualizing biology in terms of molecules (in the sense of physical-chemistry) and then applying informatics techniques to understand and organize the information associated with these molecules on a large scale [60]. It can also be seen as all the computer techniques and computational statistics for analyzing biological data [31]. There are a few other definitions on the concept of bioinformatics in many different resources in the literature, but in

the big picture, bioinformatics is the task that provides necessary search, score, and analyze biological information by algorithms and specific tools from data sets that are accumulated and curated by experts in order to provide a powerful opportunity to improve human health [62].

Bioinformatics differs from a related field known as computational biology. Bioinformatics is limited to sequence, structural, and functional analysis of genes and genomes and their corresponding products and is often considered computational molecular biology. However, computational biology encompasses all biological areas that involve computation. For example, mathematical modeling of ecosystems, population dynamics, application of the game theory in behavioral studies, and phylogenetic construction using fossil records all employ computational tools, but do not necessarily involve biological macromolecules [99].

The ultimate goal of bioinformatics is to better understand a living cell and how it functions at the molecular level [99]. In order to reach this ultimate goal, bioinformatics is divided into two groups the first one consists of the development of efficient algorithms, tools, and database systems for handling biological problems, whereas the second part is to build applications for those tools, algorithms, and database systems [99] [7].

4.3 THE BIOLOGICAL SYSTEM

Biochemical molecules such as deoxyribonucleic acid DNA, Ribonucleic acid RNA, and protein are fundamental for cellular organization [51] [67], they are responsible for many biological processes of any given organism, by playing a regulated organism's role to control different pressures during an organism's life time.

System biology is the research field that includes the study of biological components, which may be molecules, cells, organisms, entire species, or the interactions between these components. Living systems are very complex, and their behaviour may be hard to predict. To understand them, plenty of bioinformatics tasks has been launched to deal with each behavior.

4.3.1 *Basic information*

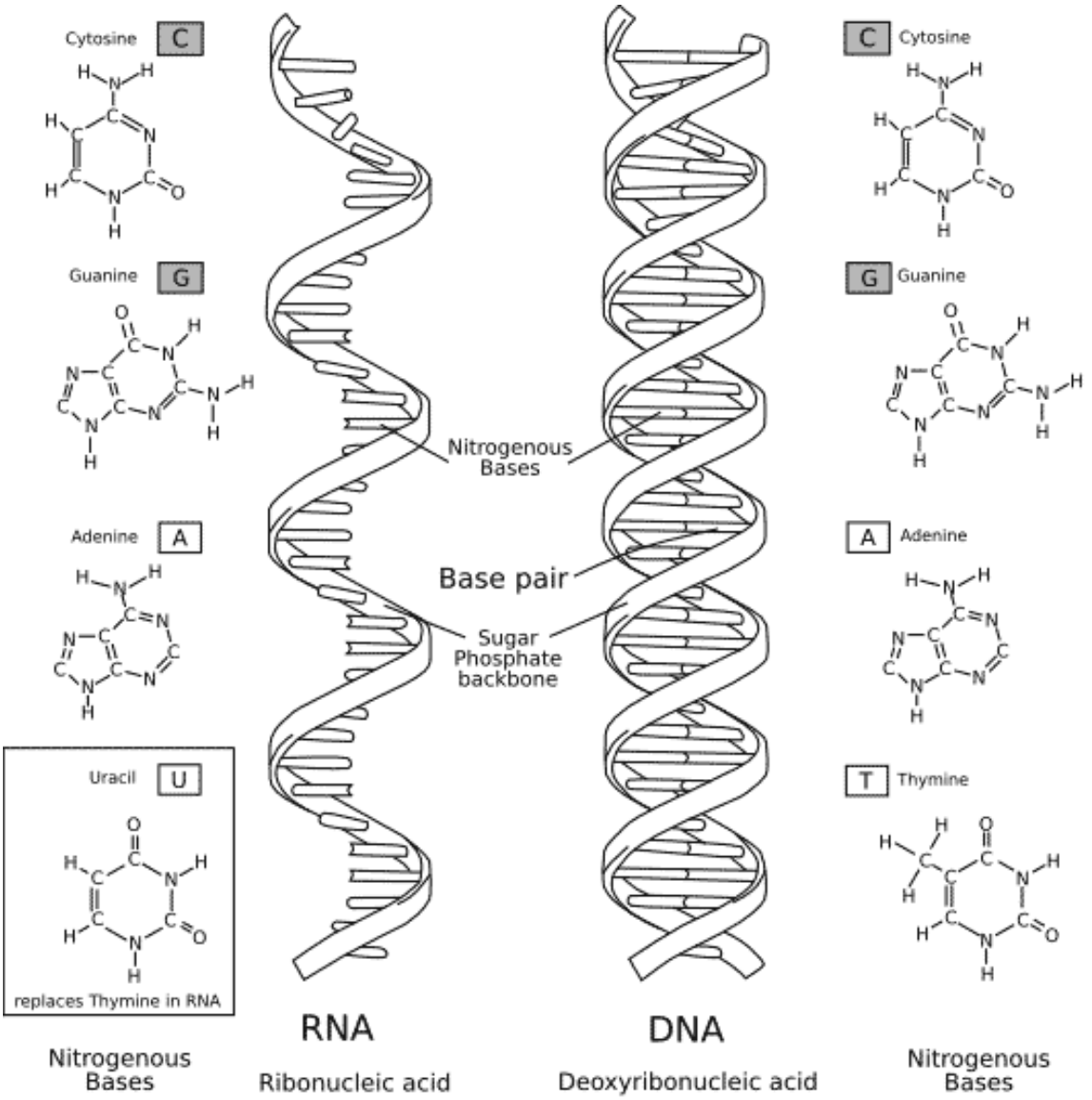
4.3.1.1 *Cell*

The cell is the smallest unit and the basic structure of any living organism, typically 1030 millionths of a metre (1030μm), contains many specialized structures called organelles, which contain many biomolecules such as proteins and nucleic acids. Organisms can be classified as unicellular (consisting of a single cell, including most bacteria) or multicellular (including plants and animals).

4.3.1.2 *Nucleic Acids: DNA Versus RNA*

DNA stands for deoxyribonucleic acid, while RNA is ribonucleic acid. DNA is like a blueprint of biological pathways that a living organism must follow to remain functional, while RNA help carry out this blueprint during organism life.

Figure 13: Structural differences between DNA and RNA



Nucleic acids are made of long chains of nucletotides that consist of nitrogenous base, a sugar moiety, and phosphodieseter connections [62]. DNA basically formed from sequence of nucleotide that is shaped as a double helix which consists of its phosphate group, five-carbon sugar, and four nitrogen-containing nucleobases: adenine, thymine, cytosine, and guanine. While RNA, formed from a single-chain. Like DNA, RNA is composed of its phosphate group, five-carbon sugar, and four nitrogen-containing nucleobases: adenine, uracil (not thymine), guanine, and cytosine. [28]

4.3.1.3 *Proteins*

A large molecule composed of one or more chains of amino acids in a specific order; the order is determined by the base sequence of nucleotides in the gene that codes for the protein. Proteins are required for the structure, function, and regulation of the body's cells, tissues, and organs; and each protein has unique functions. Examples are hormones, enzymes, and antibodies.

4.3.1.4 *Amino acid*

Any of a class of 20 molecules that are combined to form proteins in living things. The sequence of amino acids in a protein and hence protein function are determined by the genetic code.

4.3.1.5 *Domain*

A discrete portion of a protein with its own function. The combination of domains in a single protein determines its overall function.

4.3.1.6 *Motif*

A short conserved region in a protein sequence. Motifs are frequently highly conserved parts of domains.

4.3.1.7 *Profile*

A table that lists the frequencies of each amino acid in each position of protein sequence. Frequencies are calculated from multiple alignments of sequences containing a domain of interest.

4.3.1.8 *Gene*

The fundamental physical and functional unit of heredity. A gene is an ordered sequence of nucleotides located in a particular position on a particular chromosome that encodes a specific functional product.

4.3.1.9 *Codon or Genetic code*

The sequence of nucleotides, coded in triplets (codons) along the mRNA that determines the sequence of amino acids in protein synthesis. A gene's DNA sequence can be used to predict the mRNA sequence, and the genetic code can in turn be used to predict the amino acid sequence.

4.3.1.10 *Alignment*

The process of lining up two or more sequences to achieve maximal levels of identity (and conservation, in the case of amino acid sequences) for the purpose of assessing the degree of similarity and the possibility of homology.

4.3.1.11 *Global Alignment*

The alignment of two nucleic acid or protein sequences over their entire length.

4.3.1.12 *Local Alignment*

The alignment of some portion of two nucleic acid or protein sequences.

4.3.1.13 *Gaps*

A position in an alignment that represents a deletion within one sequence relative to another. Gap penalties are requirements for alignment algorithms in order to reduce excessively-gapped regions. Gaps in alignments represent insertions that usually occur in protruding loops or beta-bulges within protein structures.

4.3.2 *Biological database*

A database system is a collection of information organized in such a way that a computer program can quickly select desired pieces of data. [8] In the other hand, a biological database is organized by biology roles usually associated with biological tools to update, query, and retrieve components of the data stored within the system [3].

The task of collecting biological sequences and storing them into a data support goes back to the early 1980s under a project named the Human Genome Project [31].

In the last decade, a lot of bioinformatics work was focused on improving database systems, the following are some of the most-known biological databases:

1. **GenBank** (Genetic Sequence Databank) is one of the fastest growing repositories of known genetic sequences. It has a flat file structure that is an ASCII text file, readable by both humans and computers. In addition to sequence data, GenBank files contain information like accession numbers and gene names, phylogenetic classification and references to published literature.There are approximately 191,400,000 bases and 183,000 sequences as of June 1994.

2. The **EMBL** Nucleotide Sequence Database is a comprehensive database of DNA and RNA sequences collected from the scientific literature and patent applications and directly submitted from researchers and sequencing groups. Data collection is done in collaboration with GenBank (USA) and the DNA Database of Japan (DDBJ). The database currently doubles in size every 18 months and currently (June 1994) contains nearly 2 million bases from 182,615 sequence entries.

3. **SwissProt** is a protein sequence database that provides a high level of integration with other databases and also has a very low level of redundancy (means less identical sequences are present in the database).

4. The **PROSITE** dictionary of sites and patterns in proteins prepared by Amos Bairoch at the University of Geneva.

There are more of biological databases than we cited in this short list, some of these databases are public repositories such as GenBank, other are private databases.

4.4 BIOINFORMATICS TASKS

Biological problems can be countless due to the importance of biology which is the understanding of the nature of life, that includes understanding and im-

proving human health. Bioinformatics was invented to deal with those problems, many bioinformatics tasks are initiated for each biology problem.

In the following we describe a few bioinformatics tasks that were recently very active.

4.4.1 *Data collection*

Because bioinformatics is driven by the outpouring of massive genomics data, data collection became an important task in bioinformatics aimed to invent and improve the tools to transform biological sequences from a biologist's perspective to a sophisticated readable sequence stored into a data server.

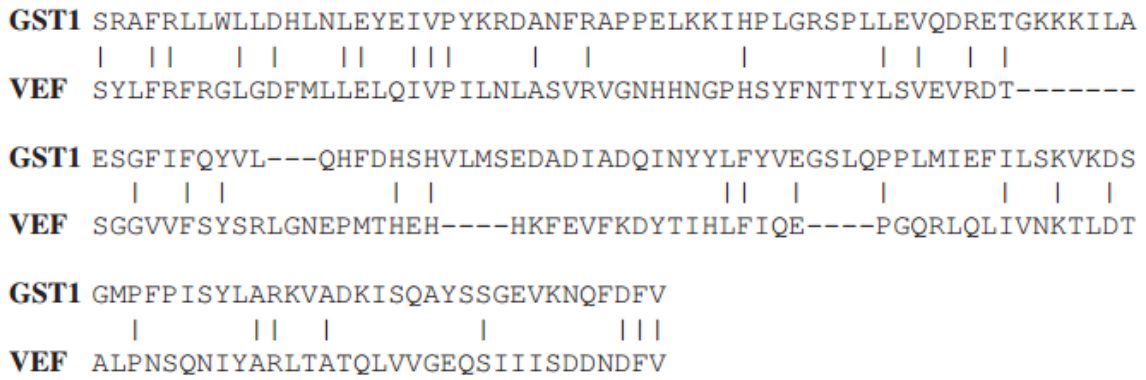
4.4.2 *Motif discovery*

Motif is a short sequence that represents an expression of characterising biological function. Motif discovery is the process of extracting motifs from biological sequences such as DNA, RNA, and Protein.

4.4.3 *Sequence alignment*

Alignment is a model used by biologists for bringing up sequence similarity [16], For example, an alignment between two sequences is shown in Figure 14

Figure 14: Sequence alignment between two sequences



Besides these tasks, there are plenty more bioinformatics tasks that have been initiated to deal with other kinds of biological problems.

4.5 BIOINFORMATICS APPLICATIONS

There are many application of bioinformatics in various fields, which be reasonable due to the bioinformatics tasks that have been established. For example if we take the field of medicine we find that bioinformatics can search for possible diseases through analysing every genetic component of a set of diseases and comparing that with the patient's profile

Personalizing medicine is another application of bioinformatics in the context of medicine, which means that doctors will be able to analyse a patient's genetic profile and prescribe the best available drug therapy and dosage

Gene therapy, drug development, waste clean-up and many other bioinformatics applications can be found for the purpose of improving human health.

Part III

CONTRIBUTION PART

In this part we propose a new algorithm called MEPda (Motif Extraction algorithm based on Pushdown automata) in order to solve the problem of finding patterns containing loops. These loop patterns or loop motifs are very known and used in many domains, especially in mathematics and bioinformatics. MEPda meant to find these kinds of patterns by using pushdown automata as a mechanism of matching process alongside with a counter to verify the acceptance length of loop in an optimistic way of looking.

5

MEPDA ALGORITHM

5.1 INTRODUCTION

In computer science, text remains the main form of representing and exchanging data in a simpler and more efficient way in a range of areas [63] [35]. Text can be defined as a human-readable sequence of characters and is usually distinguished from non-character encoded data, such as graphics or images [76].

Pattern matching is a very important task in text processing, aimed to find a piece of text called pattern P = p1p2 ... pm, in a given text T = t1t2 ... tn where n is always larger than m, and pi, ti are symbols of the alphabet [63]. The task itself can be divided into two ways of matching, one called exact pattern matching which aims to locate the pattern P = p1p2 ... pm in the text T = t1t2 ... tn where all pi are identical to ti. The other way called approximate pattern matching, also called pattern matching that allows errors. The main purpose of this way is to find a pattern in a text where one or both of them have been exposed to some kind of corruption. Two simple examples for approximate pattern matching are finding DNA subsequences after possible mutations, and finding words in a normal text where there have been spelling errors [64].

The problem of text processing, specifically that of pattern matching, goes back at least to the 1970s, where the problem appear in many domains [63] [64]. By that time only three large application areas motivated the field of pattern matching [64], which are computational biology [80] [65] [92], signal processing [57] [89], and text retrieval [61] [90] [66]. Thereafter, the field of pattern matching became a very active research filed, more and more domains have solved many problems using algorithms depend mainly on pattern matching, such as, network security [26] [5] [73], image processing [18], spelling correction, web search engine, and speech analysis.

The evolution of biological data in terms of size or nature, caused by recording huge data by biologists using many different techniques [31], has caused

an appearance of a new concept on the research area: bioinformatics. In short, bioinformatics is conceptualizing biology in terms of molecules (in the sense of physical-chemistry) and then applying informatics techniques to understand and organize the information associated with these molecules on a large scale [60]. It can also be seen as all the computer techniques and computational statistics for analyzing biological data [31]. There are few other definitions on the concept of bioinformatics in many different resources in the literature, but in the big picture, bioinformatics is the task that provides necessary search, score, and analyze biological information by algorithms and specific tools from data sets that are accumulated and curated by experts in order to provide a powerful opportunity to improve human health [62].

Due to the enormous number of biology challenges that rose in the last few decades, several bioinformatics tasks have been lunched to deal with each challenge. One of the most important tasks on those challenges is the discovery of motifs from biological sequences in order to define the function or the family of biochemical molecules (DNA, RNA, and Protein). The nature of motif discovery task, that is depending on analyzing information of data sets represented on text format has made the task a target field of implemented and improved pattern matching algorithms to adept at addressing a specific class of pattern on such data.

The first step of discovering motif is to choose a model in order to represent the searched pattern, so far. There are three common model matrix representation, regular grammar representation, and string representation [56].

In the matrix representation model, motifs are represented by Position Weight Matrices (PWMs) or Position Specific Scoring Matrices (PSSMs) (e.g., W-AlignACE [19], DEMGD [9], Kahara J et al [49]). Algorithms based on regular grammar representation assumes that all the patterns are satisfying a set of rules, based on this view the mechanism of those algorithms search for sequences that meet the rules of pattern to be considered as a founded pattern (e.g., ono et all [69], EXMotif [101], SMotif [102]). Algorithms using string representation such as (Francis et al [20], MoTeX-II [71]).

Recent research in the field of pattern matching in the last decade shows that automata based approaches would be very simple and very useful tools for understanding and solving many pattern matching problems [63]. Based on this

perspective, we designed MEPda to meet the demand for loop-motif representation by using pushdown automata as a mechanism of matching process alongside with a counter to verify the accepted length of loop in an optimistic way of looking.

5.2 DEFINITIONS AND TOOLS

5.2.1 *Basic definitions*

In computer science, an alphabet is a nonempty finite set whose elements are called symbols. We denoted an alphabet by $\sum$. A simple example of an alphabet that can hold DNA symbols is A, G, C, T.

A string is a finite series of symbols from a given alphabet. We denote the empty string by ε; a string that has no symbols from the given alphabet. The set of all the strings in the alphabet $\sum$ is denoted by $\sum^*$, and the set of all the strings in the alphabet $\sum$ except the empty string ε is denoted by $\sum^+$. For the sake of simplification $\sum^* = \sum^+ \cup \{\varepsilon\}$.

The length of a string x is usually the number of symbols from $\sum$ in it, and it's denoted by $|x|$. It holds that $|x| \geqslant 0$ and $|\varepsilon| = 0$ where $|x| \in N$. The length of a string is very useful to get a symbol from a specific position in x. Thus, we denote by x[i], for i = 0, 1,..., $|x-1|$, the symbol at index i of x with the knowledge that the indices starts from 0. Based on this, we can write string x as follows:

$$x = x[0]x[1]...x[|x|-1] \tag{1}$$

Frequent use of strings in the area of computer science in general and in formal language theory in particular produced a variety of string functions in order to fulfil those areas' requirements. Some commonly used functions are string concatenation, string reversing, string substitution, string projection, and string matching. However, the most relevant function to our paper is string equality. It is aimed to compare two strings x, y and tell us if the string x is the same as string y or not, thus the equality between x and y would be x = y. For more simplification:

$$|x| = |y| \text{ if } x[i] = y[i] \text{ for } i = 0, 1, ..., |x| - 1 \tag{2}$$

Based on this simple function, an important class of string algorithms has risen from litterateurs, it is called string matching or pattern matching. String equality compares two strings with the same length, while pattern matching compares a string y of length L_y with part of a string x of length L_x. and that could be explain as follows:

$$|x| \geqslant |y| \text{ and } x[i] = y[j] \text{ for } j = 0, 1, ..., |y| - 1 \tag{3}$$

Pattern matching can be defined as a Boolean function, due to the output of its algorithms. If the output is true that means that the pattern y is included in string x and if the output is false that means that the pattern y is not included in the string x. Sometimes the output may be complex, such as a probabilistic value. This type of algorithms is called approximate pattern matching, also known as pattern matching that allows errors. As we said earlier in the introduction, computational biology is one of the research fields that motivated the research on pattern matching algorithms the most, due to the importance of this task in biology. Patterns can be found in biological sequences such as deoxyribonucleic acid (DNA), Ribonucleic acid (RNA), and protein. These patterns, sometime called motifs, are presumed to have a biological function. Locating them in biological sequences using pattern matching techniques would help solve many biological problems. This process of pattern matching when it is applied to computational biology is called motif discovery.

5.2.2 *Motif discovery algorithms*

There are plenty of research fields that share the same likeness with motif discovery, such as Path Mining, Traversal Pattern Discovery, Sequence Mining, Frequents Item Set Mining... etc. this resemblance is because they are all inherited from the one top re-search domain which is pattern recognition. This diversity in the field of pattern recognition is due to the different challenges in each area, if

we talk about path mining we find that almost all path mining algorithms deal with graph data representation and this can be found often in mobile computing environment (MCE) as an example. MCE has features like high mobility, frequent disconnections, and lack of resources, path mining algorithms here can be very useful to avoid such faults. Frequents item set mining is intended to discovering interesting relations between variables in large databases, the famous application of this field is supermarket customer behavior which is customer purchase list. Motif discovery algorithms deal with extracting pattern from a given input string using many techniques include string techniques As there is a growing interest in regulatory elements that can lead to understanding some virus functions, detecting new drug, classifying spices, or to get many other helpful new knowledge of biology, the researchers have developed many algorithms in order to discover this small part of biochemical molecules. Several authors have proposed frameworks for classifying motif discovery algorithms due to the complexity of these algorithms caused by the large differences between them.

5.2.2.1 *Building the algorithm*

The field of motif discovery brings together researchers from several domains. To build algorithms, you have to know about at least three domains. First, biology, through knowing how the system of transcriptional regulation works, coding region, and regulatory binding site...etc. Statistics is the second domain, where building discovery algorithms depends on knowing how to use training data and cleaning noise. The last, and most important domain, is informatics, by knowing how to design and build efficient algorithms taking into count all techniques from previous domains.

Brazma et al [11] have defined some criteria that can interfere in building the kernel of the algorithm:

1. Training set, is the backbone of any discovery algorithm. Choosing the right type of data can make a huge effect on the results that the algorithm delivers.

2. Pattern model can be as simple as a Boolean type (return true or false), and it can also be more complicated, such as a model returning probabilistic results.

3. Design the algorithm. For this purpose, we are inspired by the work of Brazma et al [11], who concluded that the last criterion in building a motif discovery algorithm is designing the algorithm taking into account the two previous criteria, by carrying training set on a sophisticated data structure and developing a matching process for the pattern model.

Measuring the performance of a motif discovery algorithm is an important task after developing it, due to the sensitivity of applications that implement these kinds of algorithms. There are different ranking methods depending on the pattern model and the type of training set. The quality of the training set may have some influence on the ranking process because, in reality, the sequences coming from biological experiments may contain errors [11].

5.2.2.2 *Classification of algorithms*

Board range of computational methods are described in different vocabulary and have different focus on the context of motif discovery. This diversity has caused a difficulty for researchers to spot similarities as well as differences between methods, leading to the appearance of many frameworks to classify motif discovery algorithms [78]. However, there are some known classifications in the literature of motif discovery that gives us a great perspective on how classification of these methods can be done.

Das et al [24] classifies the available motif discovery algorithms into three major classes. There are those based on promoter sequences of co-regulated genes, others are based on phylogenetic footprinting, and finally those based on merging the first two methods. These three classes result from differentiating motif discovery algorithms based on what kind of data are used. Another classification can be found in which the functionality of the pattern model is the criteria, this classification brings two major classes:

1. Word-based algorithms (WB): preferred for discovering short motifs, especially when implemented with suitable data structures, but can still be a bad choice for extracting transcription factor motifs that often have several weakly constrained positions [88]. Many of WB algorithms are using regular expressions (RE) in their matching process, because RE is a powerful notational algebra describing strings and sequences [43].

Some widely used Word-Based Algorithms are shown in reverse chronological order in Table 3.

Algorithm	Principle	Date	Reference
MaMF	Enumeration	2006	[45]
WordSpy	Dictionary	2005	[91]
DMotif	Enumeration	2003	[86]
MOPAC	Enumeration	2003	[37]
MITRA	Prefix tree	2003	[32]

Table 3: Word-base motif discovery algorithm

2. Probabilistic algorithms: As the probabilistic motif is hard to understand and explain, a Probabilistic algorithm (PA) describes this type of motif with a position weight matrix (PWM) [12]. This type of algorithms is preferred for finding a longer motif compared to word-based algorithms.

   Some widely used Probabilistic Algorithms are shown in reverse chronological order in Table 4

Algorithm	Principle	Date	Reference
ALSE	Expectation maximization	2006	[56]
GibbsST	Gibbs sampling	2006	[84]
PhyloGibbs	Gibbs sampling	2005	[85]
GIMF	Expectation maximization	2005	[72]
QuickScore	Consensus	2004	[75]

Table 4: Probabilistic motif discovery algorithm

Brazma et al [11], In addition to his definition of some criteria that can interfere in building the kernel of the motif discovery algorithm, he has also categorized them with respect to whether they use explicit negative sequence sets or not, expressiveness of the pattern models, whether patterns are deterministic or statistical, and whether the algorithms are pattern driven or sequence driven.

Three classification approaches have been described, and many others can be proposed in the literature of motif discovery algorithms. Establishing a standardized framework for classifying and testing motif discovery algorithms would be an important contribution to the field, given the fact that this could make it easy for researchers to identify the choices that have to be made when a new motif discovery algorithm is being developed [78].

5.2.3 *Automata theory*

The theory of automaton has been designed a long time ago in the context of circuit verification [22], and has since been helpful in a broad range of domains [41]. It is a theoretical branch of computer science on which computer scientists are able to understand how machines compute functions and solve problems.

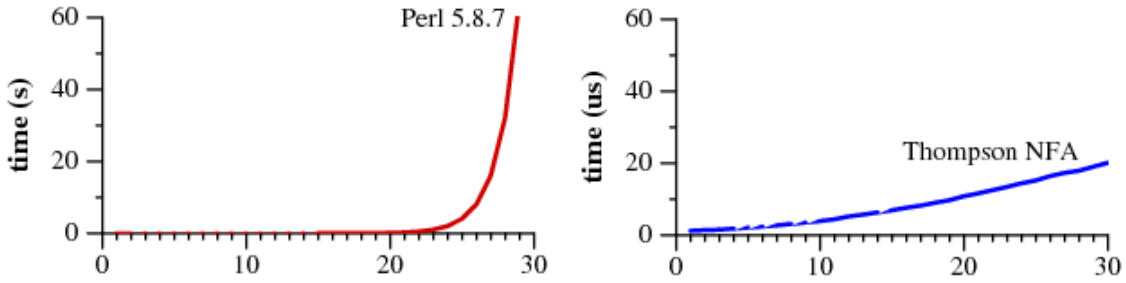
Almost every type of automata theory has the following three main components [6]:

1. Inputs: some kind of sequence of symbols that need to be analyzed.
2. States: whose role is to build a bridge between inputs and outputs.
3. Outputs: or acceptance states, once the automaton reaches an accepting state, it accepts the input.

There are four major families of automata: the Finite-State Machine is the simplest automaton; Pushdown Automata, which have an extra memory in the form of a stack; Linear Bounded Automata; and finally the Turing Machine, which is the most complex one [6] .

Structural representation is an important notation that is not automaton like but plays an important role in the study of automata and their application [46]. Regular expressions (regex) are used as a structure of data to search and manipulate text based on a pattern, although a large number of popular tools and languages are based on it to match strings. Regex is simple on implantation point of view and powerful for matching process but it is slow against finite-state machine for matching process.

Figure 15: Time to match $a?^n a^n$ against a^n



[23], shows the time required to check whether $a?^n a^n$ matches against an using two approaches. The first represents nowday Regex engine used not just in Perl but in many popular languages and tools, while the second approach labeled Thompson NFA uses a finite stat machine in the process of matching. Thompson NFA requires twenty microseconds to match a 29-character string while Perl requires over sixty seconds to match the same string.

Finite-state automata can be defined as a set of states (initial and final) and a set of rules. Here's a mathematical equation that represents it:

$$A = (Q, F, Q_f, \triangle) \tag{4}$$

Where Q is a set of states, Q_f is a set of final states and $\triangle$ is a set of transition rules and F is the input alphabet. A given string is accepted on the NFA if it can be produced from q (initial state) using rules from $\triangle$.

5.2.3.1 *Push-down Automata*

The notion of push-down automata has been introduced by Ginsburg's [38], a PDA is an automaton [30] which has a stack, a kind of simple memory in which it can store information in a last-in-first-out fashion [54].

A formal definition of PDA can be seen as a seventuple [41]:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F) \tag{5}$$

1. Q : finite set of states;

2. Σ : finite input alphabet;
3. Γ : finite alphabet of push-down symbols;
4. δ : mapping $Q * (\Sigma \cup \{\epsilon\}) * \Gamma \rightarrow 2^{Q*\Gamma^*}$ transition function;
5. $q_0 \in Q$: starting/initial state;
6. $Z_0 \in \Gamma$: start symbol on the push-down;
7. $F \subseteq Q$: set of final states;

Unlike NFA, PDA can handle strings in which some boundaries between two symbols on the input string are found. This is achieved thanks to the memory feature of the automata.

5.3 LOOP-MOTIF PROBLEM

Languages can be defined as a set of strings in which string operations such as string matching can be useful tools for solving some language problems. There are two kinds of languages, regular and non-regular language.

We say L is a regular language if L can be expressed using regular expressions or can be recognized by a finite automaton.

On the other hand, non-regular languages cannot be recognized by a finite automata and neither can regular expressions handle these kinds of languages, because all non-regular languages include nested structures in their sets of strings.

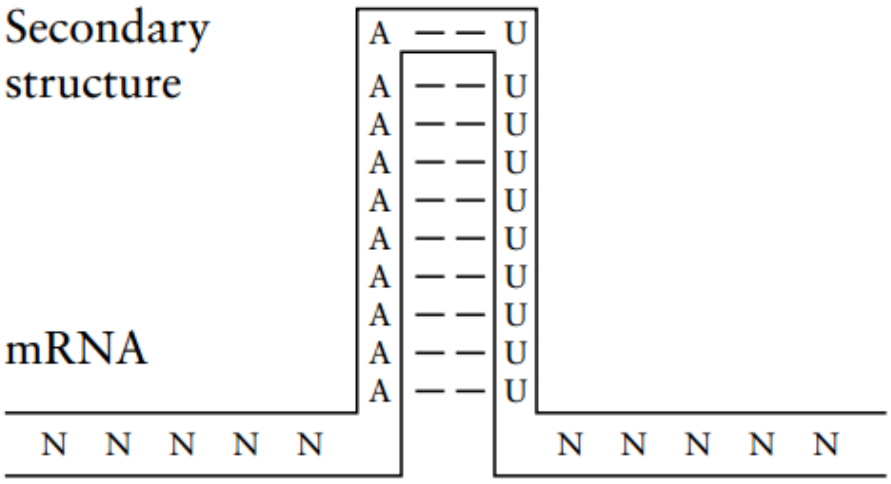
A simple example of a language that is not regular is the set of strings:

$$L = \{a^n b^n | n \geqslant 0\} \tag{6}$$

Here, regular expressions and finite automata are the wrong tools for recognizing this language, because neither of them has any memory that tells what has occurred earlier in the string. Therefore, it's impossible to remember the exact number of a's in order to check if it's the same number of b's.

Another example in the subject of computational biology is illustrated in Figure 16.

Figure 16: Looped RNA structure with equal numbers of complementary base



Despite the power of finite automata they are still not adequate for coping with certain nested structures found in biosequences, many biomolecule components contain loops between nucleotides, for example RNA sequences sometimes contain loops in order to form a double strand structure, Figure 16 describes a secondary structure of "NNNNNAAAAAAAAAAUUUUUUUUUUNNNNN" RNA sequence.

Regular expressions and finite automata are not powerful enough to accept only those strings specified by the language for these exact loops. In our example the language is:

$$L = \{A^n U^n | C^n G^n, n \geqslant 1\} \tag{7}$$

That is, while a finite automaton will accept strings of the form "AAAUUU" and "CCCGGG" it will also accept, incorrectly, strings of the form "AUUU" and "CCCCG" [31]. This mismatch happens because the matching process of finite automata works by encountering the symbol of input string by the current state of the automaton and moving on to the next state without keeping track of what has occurred earlier in the string.

This is a non-regular language problem, and there are some types of automata that can deal with this kind of problem such as pushdown automata. In this example, PDA makes sure that only those strings that have an equal number

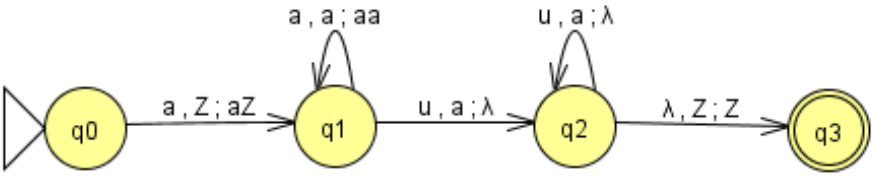
of A followed by an equal number of U (same with C and G) are accepted, by incrementing a counter for each time the symbol A or C is encountered and decrementing the value of the same counter for each time the symbol U or G is encountered. [31].

Problem of loop-motif: Indeed, PDAs are designed to represent loops in a correct way, but they can't check the exact loop that we need. The following example will describe the problem. Consider the following RNA tertiary motif sequence:

AAAUUU

Normally the PDA corresponding to this motif is:

Figure 17: PDA corresponding to AAAUUU



This PDA (note that Z is the initial symbol in the stack, and λ is the empty symbol in the stack) will accept all strings that contain this motif. It will also incorrectly accept strings that contain the same loop with more or less length, such as:

1. AAAAUUUU
2. AAUU
3. AU

This kind of mismatch has a significant impact on computational biology, in that biologists cannot rely on an inappropriate representation.

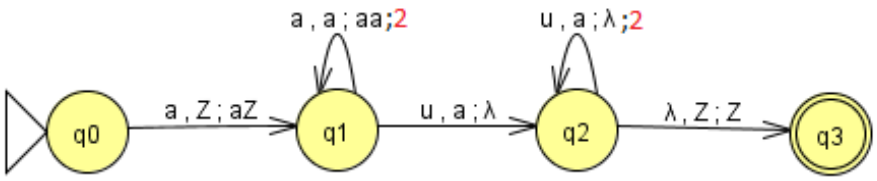
5.4 THE PROPOSED ALGORITHM (MEPDA)

Many motif sequences in biology contain loops in their structure definition, especially when it comes to those motifs that are describing secondary or tertiary structures. Exploiting the power of PDA to represent loops of those motifs will help the process of motif discovery to work more efficiently.

5.4.1 *Solution for the problem of loop-motif*

At first, we proposed a solution for the problem by adding a counter to the PDA to limit memory stack in order to accept the exact symbol that matches a given motif.

Figure 18: PDA on Figure 17 with counter



The counter works in a very simple way for the moment. The example illustrated in Figure 18 describes how our PDA matches the appropriate motif with the help of this counter.

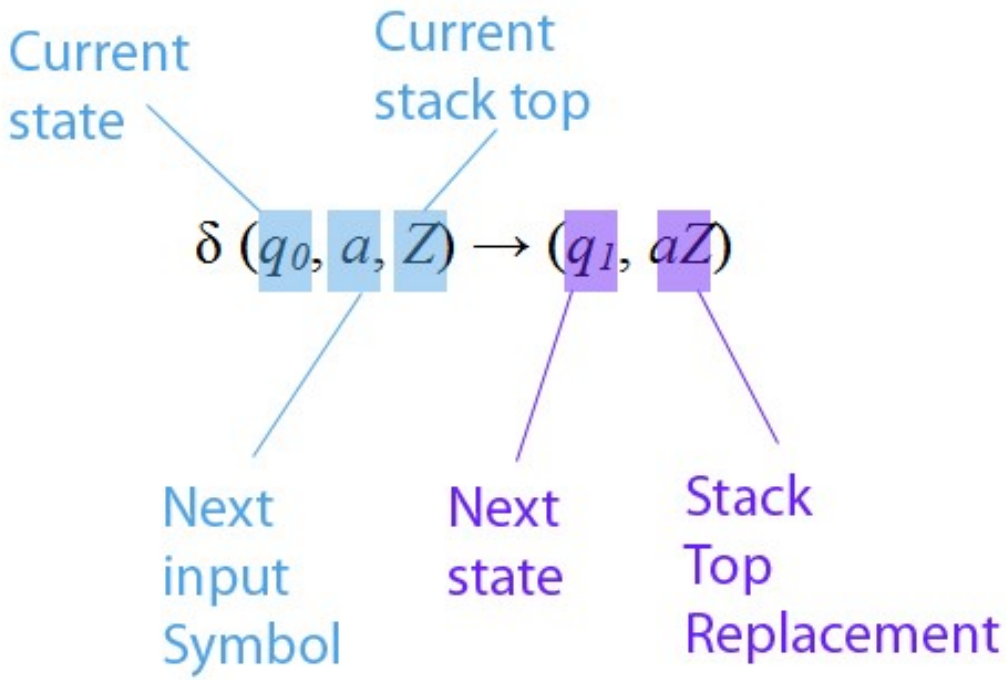
As was described in Section 5.2.3, the formal definition of a PDA can be seen as a seven-tuple. The fourth tuple, transition function, is the element responsible for moving from one state to another. Table 5 shows the set of transition functions corresponding to the PDA from Figure 17, with the possible repetitions of each transition.

Transition function	Possible repetitions
$\delta(q_0, a, Z) \rightarrow (q_1, aZ)$	1
$\delta(q_1, a, a) \rightarrow (q_1, aa)$	*
$\delta(q_1, u, a) \rightarrow (q_2, \epsilon)$	1
$\delta(q_2, u, a) \rightarrow (q_2, \epsilon)$	*
$\delta(q_2, \epsilon, Z) \rightarrow (q_3, \epsilon)$	1

Table 5: Transition functions of PDA from Figure 17

First let's describe transition functions. Figure 19 illustrates the components of a transition function, which is the first column of Table 5. The second column of Table 5 shows the possible repetitions of each transition function. For example, if we take the first function, we will notice that the PDA can only use the function for only one time, contrary to the second function where the PDA can use it many times, which is represented by the asterisk symbol (*).

Figure 19: Components of transition function



In our PDA, the mapping function will be influenced by the counter which is represents the number of loops needed. For more explanation, the set of transition functions of Table 5 will be

Transition function	Possible repetitions
$\delta(q_0, a, Z) \rightarrow (q_1, aZ)$	1
$\delta(q_1, a, a) \rightarrow (q_1, aa)$	2
$\delta(q_1, u, a) \rightarrow (q_2, \epsilon)$	1
$\delta(q_2, u, a) \rightarrow (q_2, \epsilon)$	2
$\delta(q_2, \epsilon, Z) \rightarrow (q_3, \epsilon)$	1

Table 6: Transition functions of PDA from Figure 18 (including the counter)

The same set of transition functions of Table 5 is in Table 6, except that, the possible repetitions of function 2 and 4 now depend on the counter, which in our example is 2.

The matching process of the PDA will change according to the transition functions table. This means that when the process reaches the state q1, the counter ensures that there are just two more loops and moves directly to the next state. The same thing will happen with state q2.

The example shows a very simple representation of the counter (natural number N). However, the counter can be more complex (e.g. using a range).

5.4.2 *MEPda steps description*

MEPda is a motif discovery algorithm attempts to find occurrences over sequences that contain loops such as Palindromic sequence, by using a new kind of push-down automaton that implement counter for transition functions. At the first step of MEPda, the algorithms preprocess the set of motifs and the input string.

5.4.2.1 *Preprocessing of input string and set of motifs*

Because loop-motif is a complex string type, due to the existence of some boundaries between its own symbols, we distinguish six types of loop-motifs, which are described in Table 7.

The symbol • represents one part of the loop, while · represent gaps in the motif, the gaps could be any nucleotide.

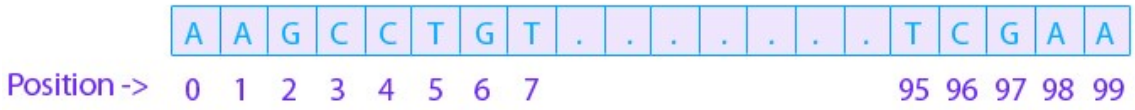
Loop-motif	Description
	MiddleGapsMiddle is a Loop-motif that starts with gaps that could be any nucleotide and the loop is split by other gaps, finally the motif ends gaps
	LeftGapsMiddle starts directly with the first part of the loop followed by gaps, which are followed by the second part of the loop. LeftGapsMiddle ends with gaps.
	The loop here is not splitted by any gaps, and LeftLeftGaps starts directly with the loop and finishes up with gaps.
	GapsRightRight similar to LeftLeftGaps except that the gaps are first and the loop follow.
	GapsMiddleMiddleGaps in this particular Loop-motif type the loop is not splitted with gaps and it's located on the middle of the motif surrounded by two gaps, one in the beginning and the other on the end.
	GapsMiddleGapsRight starts with gaps, followed by the first part of the loop, followed by more gaps, and finishes with the second part of the loop

Table 7: Loop-motif types with each type name

Many biological sequences contain these kinds of loops. We do this dichotomy of Loop-motifs in order to facilitate the matching process of the PDA.

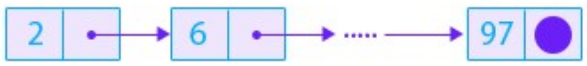
MEPda uses another kind of data structure called Linked List. A linked list is a data structure consisting of a group of nodes which together represent a sequence. The goal of using this data structure is to represent all positions of each nucleotide A, G, T, C from the input sequence, and by exploiting the quick search feature of linked list, MEPda can easily navigate through the given sequence.

Let's assume that we have the following string (DNA sequence of length equal to 99) as an input string for our algorithm:

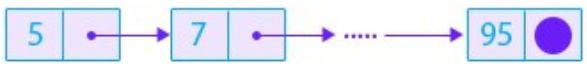


For each symbol A, G, T, C of the input string, MEPda generates a Linked List that holds all positions of that symbol.

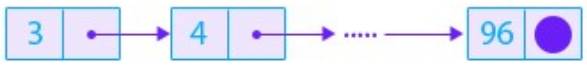
LinkedList of symbol A



LinkedList of symbol G



LinkedList of symbol T



LinkedList of symbol C

The preprocessing of the input string and the set of motifs is like an initialization step in MEPda algorithms. The result of this step is very critical for our algorithm in the next steps.

5.4.2.2 *Finding motifs*

This step represents the core of our algorithm. It attempts to find all possible motifs that are represented on a six types. Also, at this stage, MEPda no longer needs to deal directly with the input string as a full sequence of symbols, but instead uses the set of generated linked lists as input.

At first MEPda picks one motif at a time from the set of motifs, and in order to make an optimistic way of finding, MEPda looks at the first symbol of the chosen motif and takes only the Linked List that corresponds to that symbol.

Based on the positions in this picked Linked List, MEPda creates a set of substrings from each position in the List with the length that is likely to be the motif we are looking for. The following example gives more explanation:

Considering the input string T = AAGCCACTTCATTGTGATTAAC, a set of Linked Lists for each symbol will be created by the algorithm as we mentioned earlier:

1. $A = \{0, 1, 5, 10, 16, 19, 20\}$;
2. $G = \{2, 13, 15\}$;
3. $C = \{7, 8, 11, 12, 14, 17, 18\}$;
4. $T = \{3, 4, 6, 9, 21\}$;

This is the first step which is input string preprocessing. Now, suppose that we have the following motif that we try to locate it on the input string:

$$M = \text{ACTCATG}|counter = 2 \tag{8}$$

M is a MiddleGapsMiddle Loop-motif type which means that the two red symbols in the middle are the loop part. In the next step, MEPda gets the first symbol of the motif M which is A. At this point, MEPda will only focus on where

A's are located in our input string T, using the Linked List A. Starting from each position on that List, MEPda cuts a substring from T to be an input string for a PDA that represents the motif M. The length of these substrings depends on to the length of the motif and the counter. In this example length of the substrings is 9.

1. $T_1 = AAGCCACTT from position 0;$
2. $T_2 = AGCCACTTC from position 1;$
3. $T_3 = ACTTCATTG from position 5;$
4. For the rest of the positions, MEPda cannot create a substring of length 9 from them. ;

MEPda considers T_1, T_2, and T_3 as an input string for a PDA with a counter value equal to 2, which represents the motif M. With the help of this counter, MEPda will only loop once when it reaches the first T symbol, and will ensure the same thing will happen for the second part of the loop and as a result MEPda will only match the substring T_3.

T	A	A	G	C	C	**A**	**C**	**T**	**T**	**C**	**A**	**T**	**T**	**G**	T	G	A	T	T	A	A	C
M						**A**	**C**	**T²**		**C**	**A**	**T²**		**G**								

5.4.3 *MEPda pseudo-code and architecture*

The preprocessing task of the input string that creates a Linked List for each symbol is described by the following pseudo-code.

For the entire algorithm, the following pseudo-code describes how MEPda algorithm works.

The above description and pseudo-code for the MEPda algorithm can be summarized into the following architecture diagram.

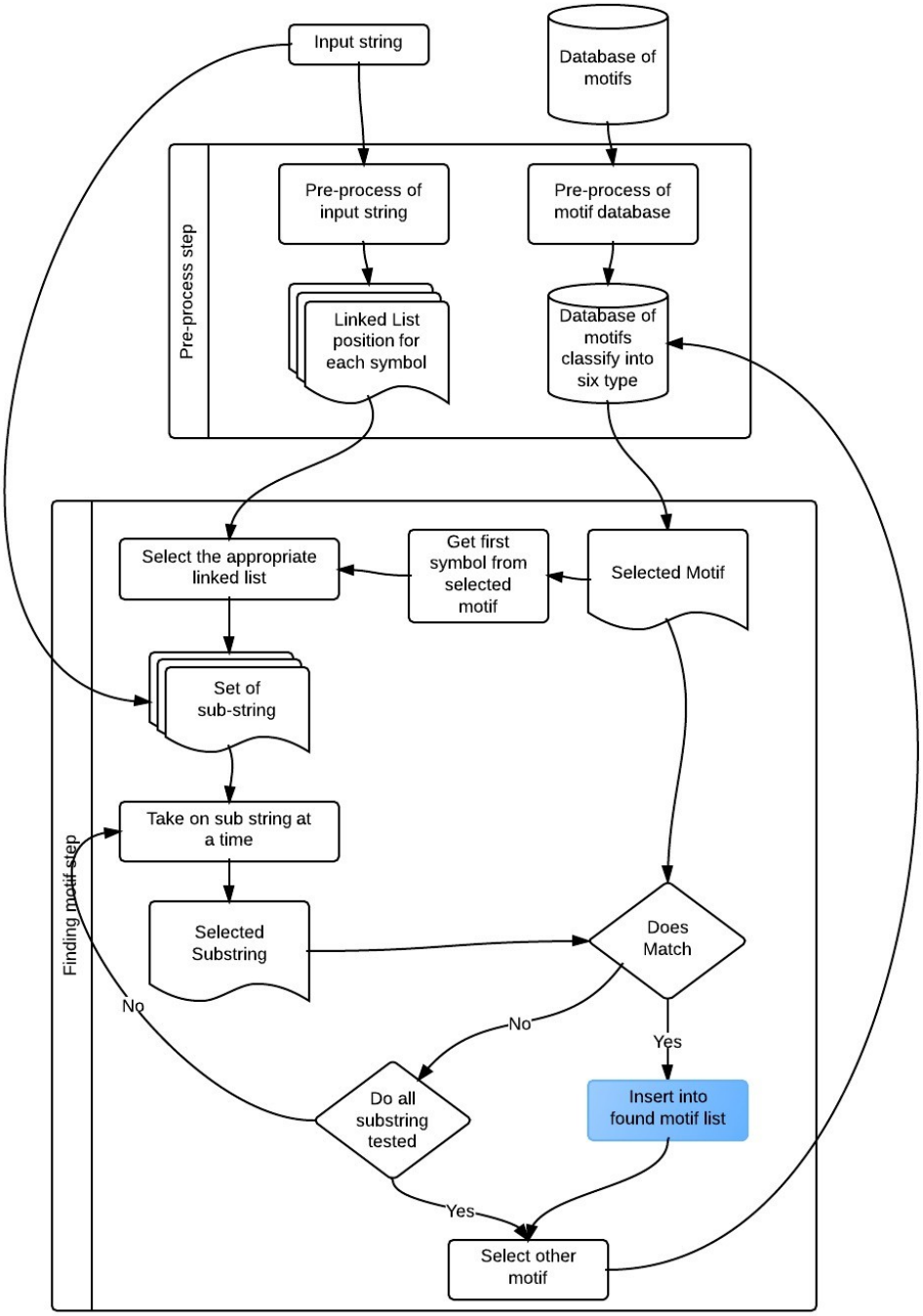
Data: Input string T
Result: Set of Linked Lists P
initialization:
$i \leftarrow 0$
$c \leftarrow''$
Create a set of empty linked lists P
while $i \leqslant$ length of T **do**
  $c \leftarrow T[i]$
  Switch c
  insert i on the appropriate linked list of the set P.
  $i \leftarrow i + 1$
end
Return P;

Algorithm 1: Preprocess of MEPda algorithm

Data: input string T, set of linked lists P, set of motifs M.
Result: set of all found motifs F.
initialization:
$c =''$
p empty linked list
Create an empty list F
while *all motif m in M* **do**
  $c =$ Get first symbol of selected motif m
  p = Select (P, c) *Select the appropriate LinkedList from P*
  F $\leftarrow$ FindMatches(p,m,T) *Try to find matches of the motif m in all positions of p in the input text T*
end
Return F;

Algorithm 2: MEPda algorithm

Figure 20: The global architecture of MEPda algorithm



5.5 EXPERIMENTAL RESULTS

We have performed comprehensive experiments on three datasets. The main objective of these experiments is to show the efficiency and the accuracy of MEPda to find loop-motifs in a given input string. All experiments are performed using *C#* as a programming language and tested on a computer with 3.2 GHz Core I3 processor and 8 GB of memory.

5.5.1 *Experiments on accuracy and runtime*

The first dataset consists of two databases, one represents protein sequences of human species collected by UniProt, while the other is a database of protein families and domains collected by Prosite.

Database	Content
UniProt (Human species)	88708 sequences represented in FASTA format. We selected only the first 100 sequence
Prosite	1308 patterns We selected only the first 69 patterns

Table 8: Contents of the first dataset

The aim of this first experiment is to prove that MEPda can find patterns of Prosite which have no loops, in the human sequences of UniProt. It also analyses the accuracy and runtime of MEPda compared to two well-known pattern matching algorithms: Aho-Corasick [4], and KMP [53].

Aho-Corasick and KMP algorithms are too aged in creation point of view, which they are proposed in 1975, and 1977 respectively. But in nowday they are still as a referenced algorithm in the context of string matching do to their acceptable runtime and accuracy for locating patterns. Some recent algorithms based on AC and KMP are ex-tension of one of this two algorithm such as Extension

of Aho-Corasick Algorithm to Detect Injection Attacks [73], And Adapting the Knuth-Morris-Pratt algorithm for pattern matching in Huffman encoded texts [83].

As expected, the three algorithms provide equally high accuracy for finding patterns in each sequence, since all the three are word based pattern matching algorithms and use the same source for patterns which is Prosite.

Table 6 also shows that MEPda is faster for matching patterns in sequences with lengths between 10 and 1000, and starts to slow down for sequences with lengths exceeding 1000. However, it is still performing reasonably well, proof being that MEPda is still better than KMP.

The results of this experiment are summarized into Table 9 and illustrated on Figure 21 and Figure 22.

Algorithm	Accuracy	Runtime (ms)				
		Length of each sequence				
		10 ~ 100	100 ~ 500	500 ~ 1000	1000 ~ 2000	> 2000
MEPda	100%	25	39	60	104	218
Aho-Corasick	100%	62	65	73	83	112
KMP	100%	31	56	113	205	424

Table 9: Statistical comparison indicating accuracy and runtime of MEPda compared to Aho-Corasick and KMP for finding patterns without loops

Figure 21: The accuracy values of MEPda, Aho-Corasick, and KMP for finding patterns of Prosite in selected sequences from UniProt

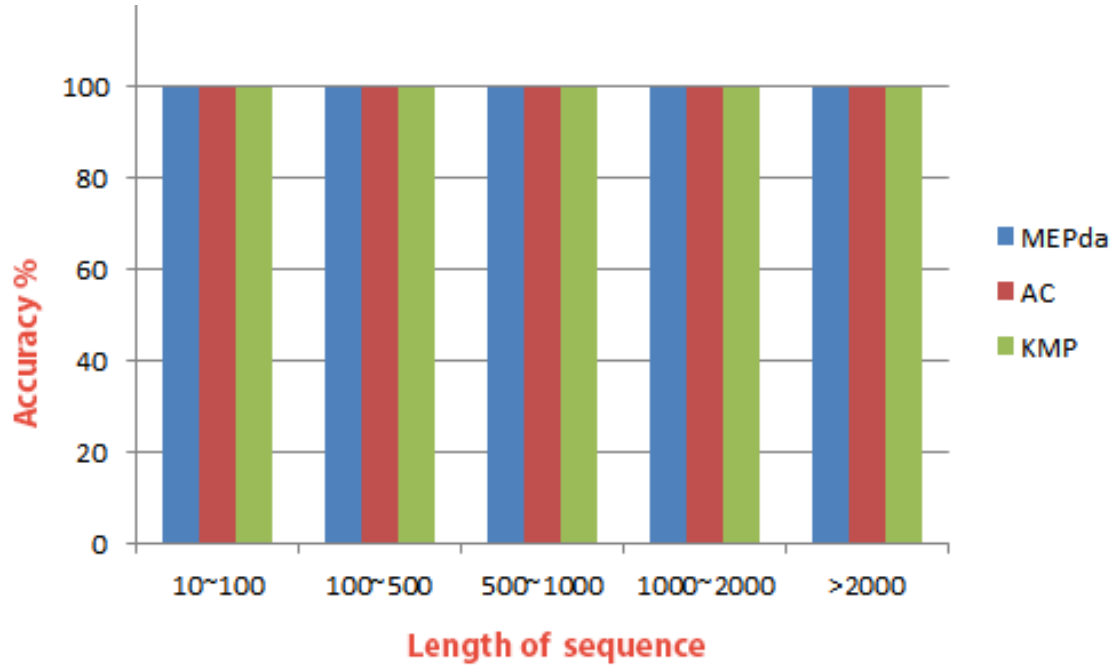
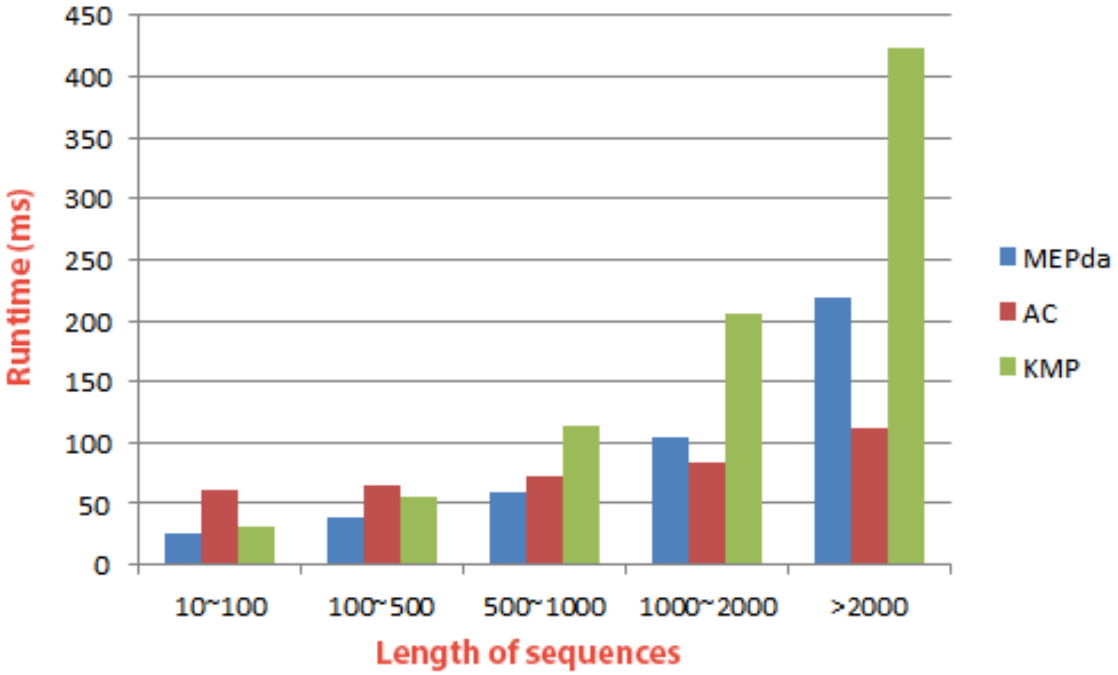


Figure 22: The runtime values of MEPda, Aho-Corasick, and KMP for finding patterns of Prosite in selected sequences from UniProt



5.5.2 *Experiments on robustness*

MEPda was designed to meet the demand of loop-motif representation as we mentioned in the beginning. In fact, we performed this experiment in order to prove that MEPda has high accuracy and high efficiency for finding patterns containing loops.

In this experiment we used the remaining datasets, the first one is an artificial database generated by C# containing more than 2000 sequences and more than 200 patterns containing loops of different types. While the second dataset is an RNA secondary structure collected by Rfam [40].

Database	Content
Artificial dataset	2000 generated sequences, and 200 generated patterns
Rfam dataset	115 sequences from vibrionaceae family, which include 110 different species.

Table 10: Contents of the second and third datasets

In the first part of this experiment, we use the second dataset in order to show the robustness of MEPda for finding patterns containing loops compared to the other algorithms.

Algorithm	Loop-motif type	Robustness	
MEPda	All types	Despite the length of loop-motif being equal to or larger than 1, MEPda can match loop-motifs with any loop length.	
		L-length=1	L-length>1
Aho corasick, KMP	Middle Gaps Middle	can match	Algorithm can only find the first gaps and the first letter from the loop
	Left Gaps Middle	can match	Algorithm can only find the first letter from the loop
	Left Left Gaps	can match	Algorithm can only find the first letter from the loop
	Gaps Right Right	can match	Algorithm can find the first gaps and the first letter from loop
	Gaps Middle Middle Gaps	can match	Algorithm can find the first gaps and the first letter from loop
	Gaps Middle Gaps Right	can match	Algorithm can find the first gaps and the first letter from loop

Table 11: The robustness of each algorithm for finding loop-motifs

Table 11 describes the robustness of each algorithm for finding loop-motifs. Apart from MEPda, all the other algorithms cannot identify loop-motifs due to the special representation that we proposed, and they stop matching at the first letter of the first loop.

In order to prove Table 11, we selected 100 sequences from the artificial dataset, each one containing between 1 and 6 loop-motifs from different types and different loop-lengths. The main criteria that influences accuracy values is finding loop-motifs with loop length equal to 1. In other words, if there are many loop-motifs of length equal to 1, the value of accuracy will increase for Aho-Corasick and KMP algorithms, because all the three algorithms share the same results of finding this specific type of motif.

Algorithm	Average Accuracy	Runtime (ms)			
		Length of each sequence			
		> 100	> 500	> 1000	> 2000
MEPda	100%	51	60	96	104
Aho-Corasick	60 ~ 70%	60	61	65	73
KMP	60 ~ 70%	33	51	92	151

Table 12: Statistical comparison indicating accuracy and runtime of MEPda compared to Aho-Corasick and KMP for finding pattern containing loops

Figure 23: The accuracy values of MEPda, Aho-Corasick, and KMP for finding loop-motifs in generated sequences

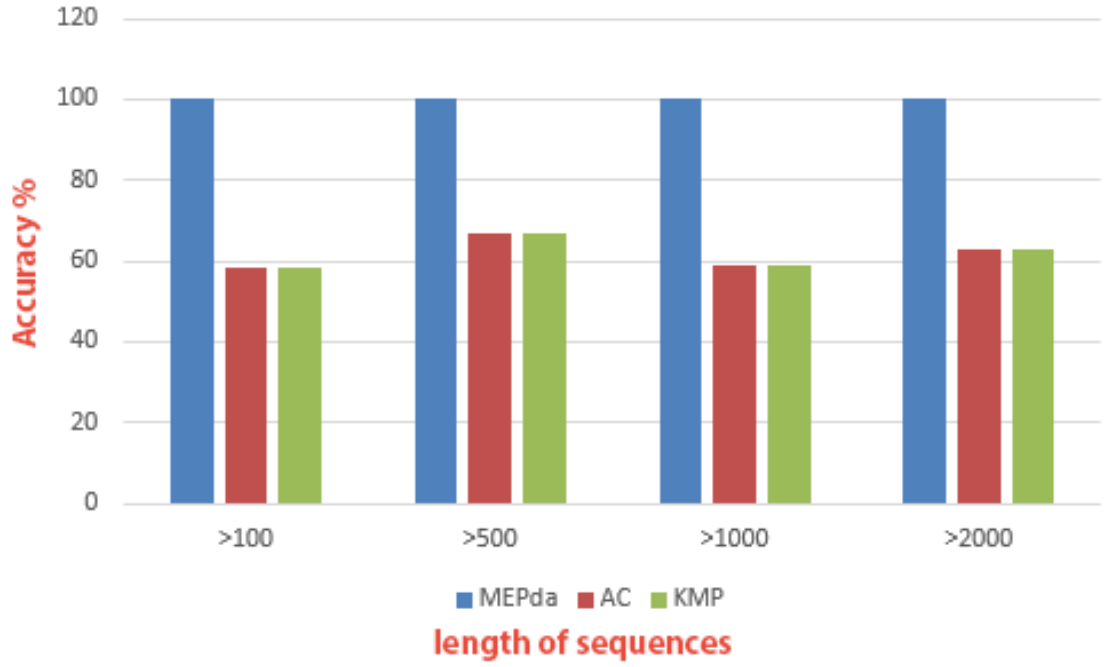
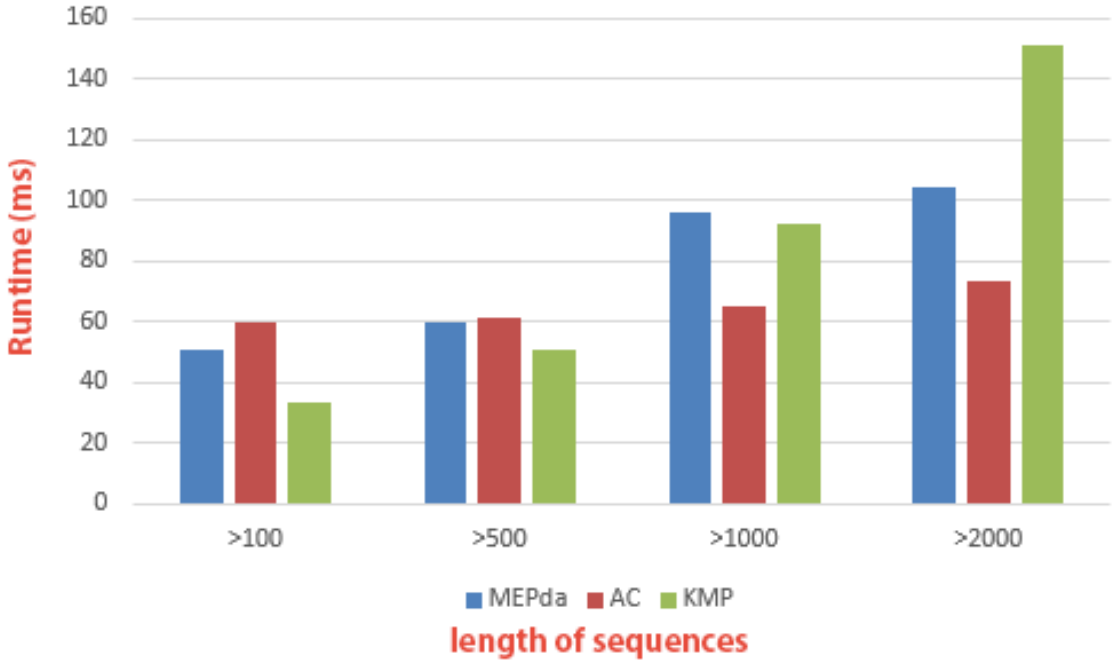


Figure 24: The runtime values of MEPda, Aho-Corasick, and KMP for finding loop-motifs in generated sequences

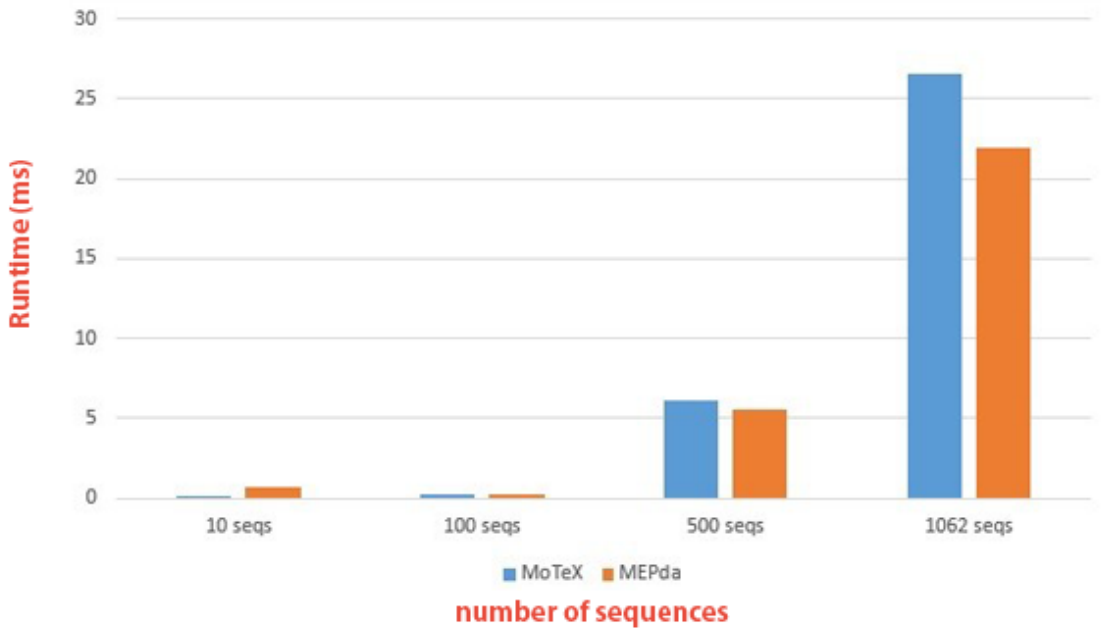


In addition, we perform another experiment between our algorithm MEPda and one of motif discovery recent algorithms, which is MoTeX [71] [29]. The experiment runs using a collection of sequences with different length and a set of motifs.

In this experiment our MEPda outperform MoTeX in accuracy point of view, as the same results when we run an accuracy test against Aho-Corasick algorithm, and KMP algorithm. This performance is due to the proposed representation of Loop-Motif in our MEPda algorithm.

The runtime of this experiment shows that our algorithm can outperform MoTeX to find motifs in a set contains over 500 sequences. However, the runtime results in finding motifs over a set contains less than 500 sequences is still promising, note that MoTeX algorithm uses 4 process in parallel in matching process, and in the other side our algorithm uses only 1 process in matching process. Figure 25 summarizes the results of this experiment.

Figure 25: The runtime values (s) of MEPda, and MoTeX for finding motifs in generated sequences



The second part of this experiment is to apply MEPda on 5 RNA sequences from Rfam, and compare runtime and accuracy of MEPda with a Regular ex-

pression based algorithm and a PDA based algorithm. Table 13 summarizes the results of this experiment.

Sequence ID	MEPda		PDA		RegEx	
	Acc	RunT	Acc	RunT	Acc	RunT
ACYU01000068.1 /17818-17635 Vibrio mimicus	66.7%	49ms	57.1%	40ms	33.3%	6ms
ACYV01000017.1 /37265-37448 Vibrio mimicus	66.7%	50ms	57.1%	39ms	33.3%	5ms
ACHZ01000017.1 /1901543-1901726 Vibrio cholerae	83.3%	47ms	71.4%	35ms	50%	4ms
AGUK01000128.1 /14553-14736 Vibrio cholerae	83.3%	48ms	71.4%	35ms	50%	4ms
AEVS01000007.1 /50190-50372 Vibrio brasiliensis	100%	30ms	50%	20ms	50%	2ms
Average	80 %	44.8ms	53,32%	33.8ms	53,32%	4.2ms

Table 13: Statistical comparison indicating accuracy and runtime of MEPda compared to PDA based algorithm and Regular expression based algorithm

There are some observations on this table that need to be explained. First of all, we can notice that the accuracy of our algorithm has been reduced due to expose some mo-tifs on mutation in nucleotides that form loop part. Nonetheless, MEPda is still highly accurate compared to other algorithms, followed by the PDA based algorithm which is capable of finding some loop motifs due to the memory property of PDA, unlike the RegEx based algorithm. Secondly, we can also see that the fastest algorithm in this ex-periment is RegEx due to the average

length of sequences on the dataset used, which does not exceed 200, and as we know RegEx is very fast when working on strings with small lengths, unlike PDA that can handle larger lengths of strings in less time. The experiment results are summarize into Figure 26, and Figure 27.

Figure 26: The accuracy values of MEPda, PDA, and RegEx for finding loop-motifs in Rfam sequences

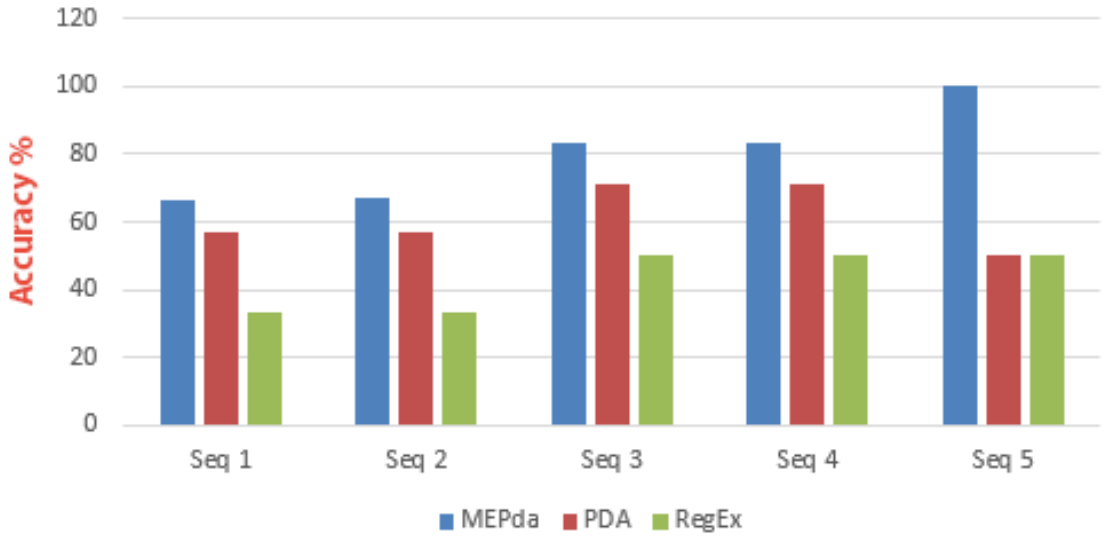
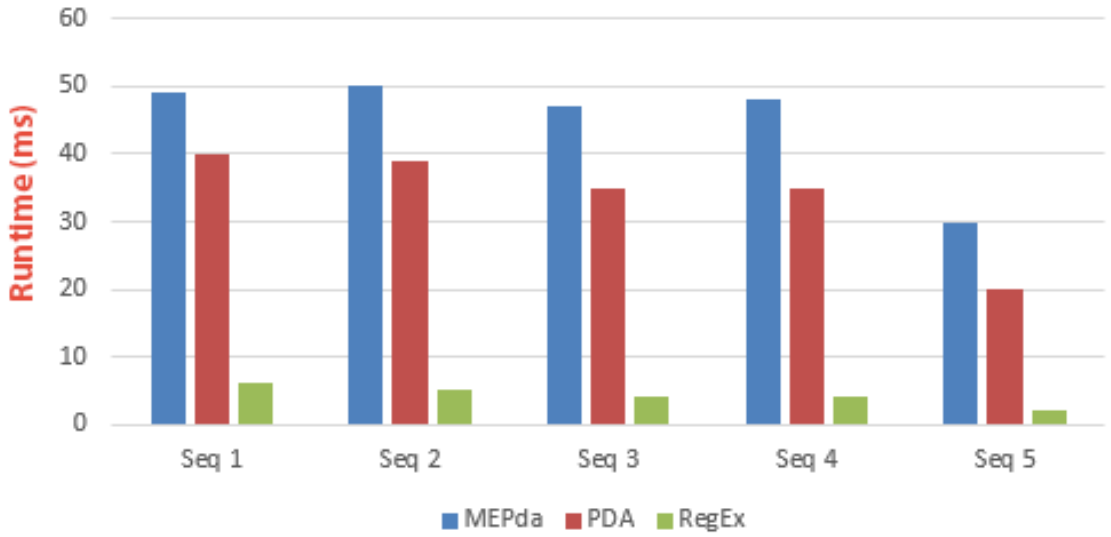
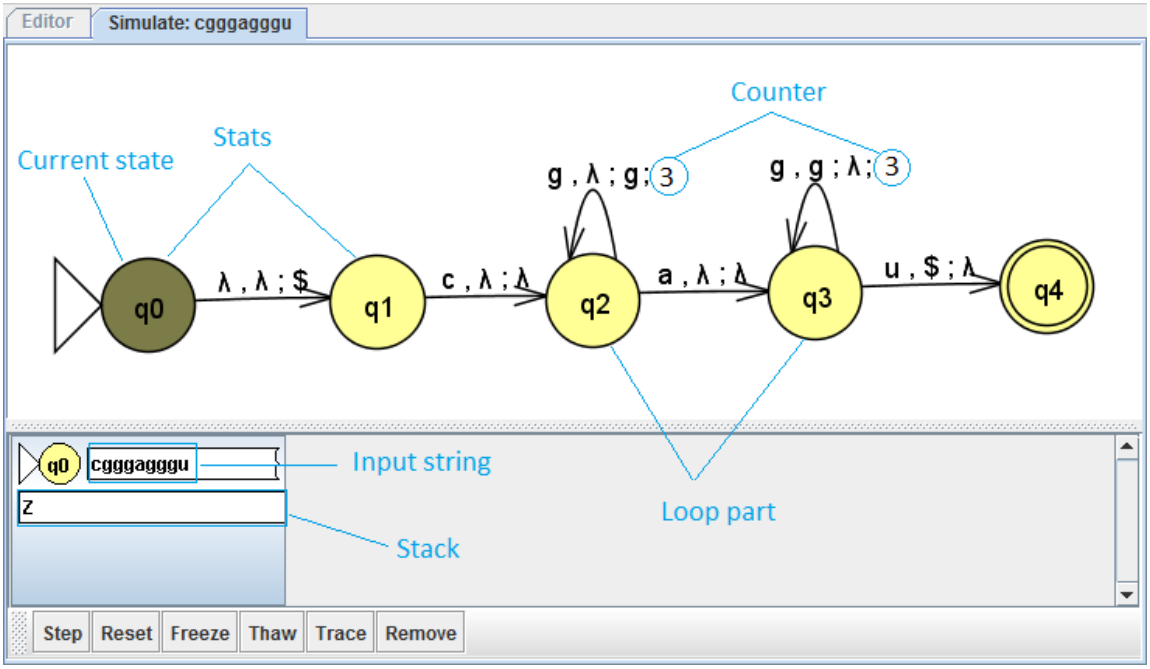


Figure 27: The runtime values of MEPda, PDA, and RegEx for finding loop-motifs in Rfam sequences

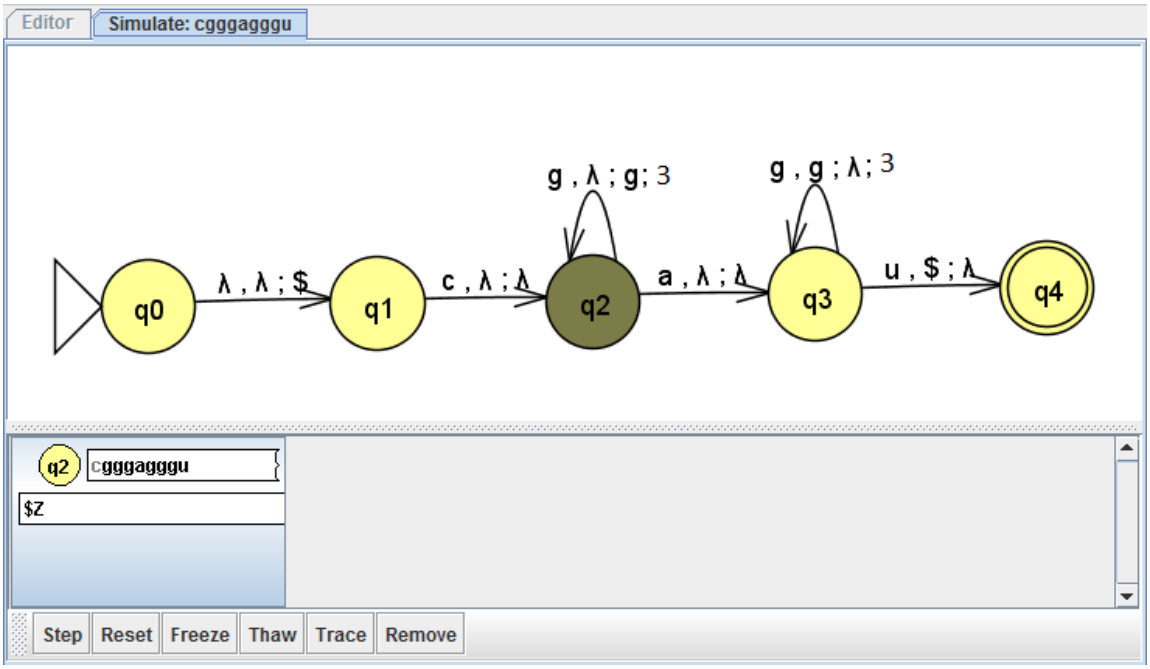


In the last part of this experiment, we provide a simulation of how MEPda works and matches patterns. The simulation was ran on JFLAP [59], one of the major visualization projects to simulate automata.

We take 2 examples, one with a positive match, while the other is negative. Example 1: the pattern is p = CG3AG3U a loop-motif with length equal to 3, and an input text, which is a part of sequence seq = ...CGGGAGGGU...

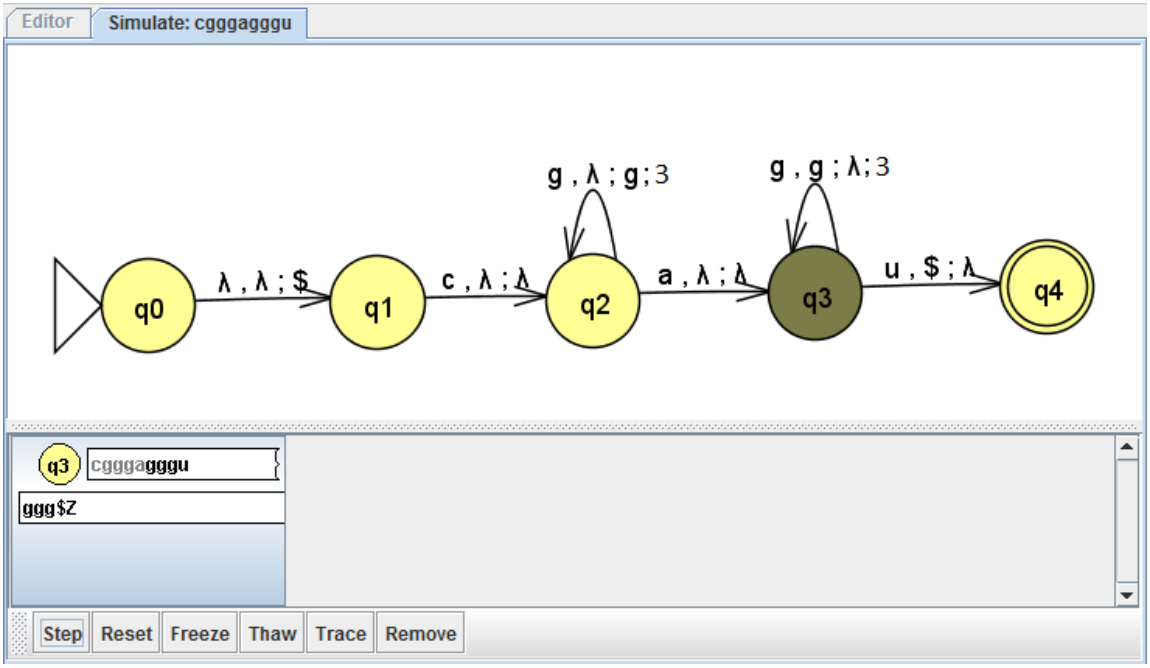


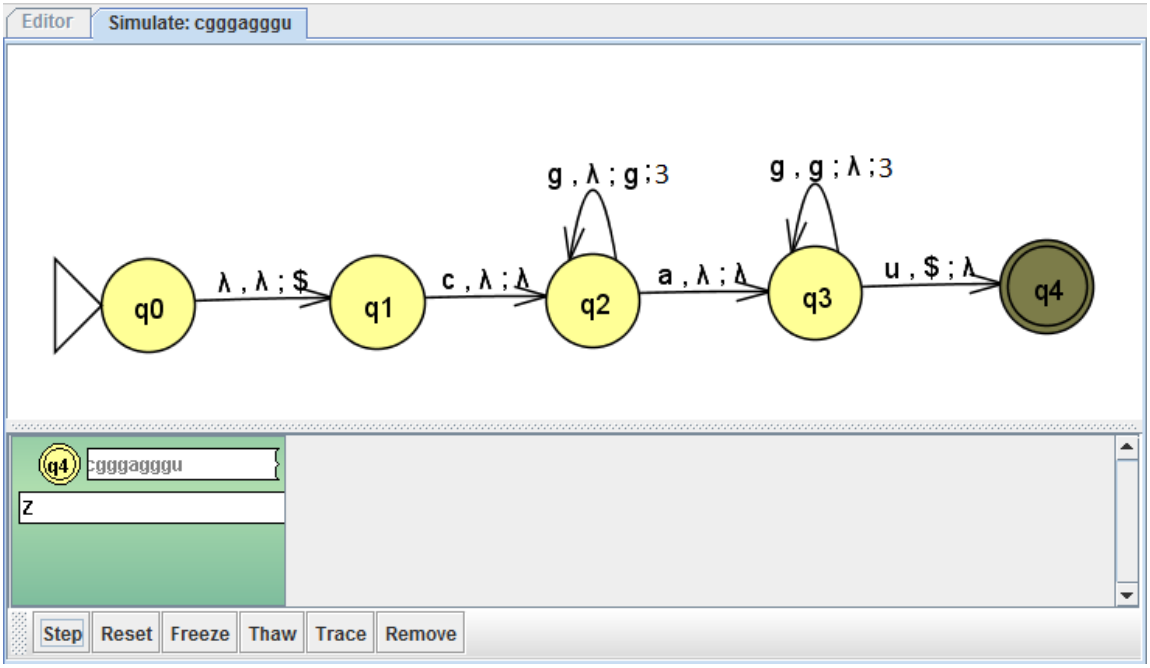
The simulation starts with the initial state q0 with empty stack, and moving forward to q1 by putting the symbol dollar in the head of stack. When the simulation reaches the state q1 the input string must start with the symbol c in order to move to q2. There is no pushing or popping from stack at this stage.



In this step, the counter (equal to 3) makes sure the state q2 will loop only 3 times, and pushing the symbol g to the stack each time, and move forward to the state q3 if the next symbol from input string is a.

In the same way, the state q3 will loop 3 times and for each time will pop one g from the stack, and move to q4 if next symbol from input string is u, in this step MEPDA will pop the symbol dollar and reach the final state.

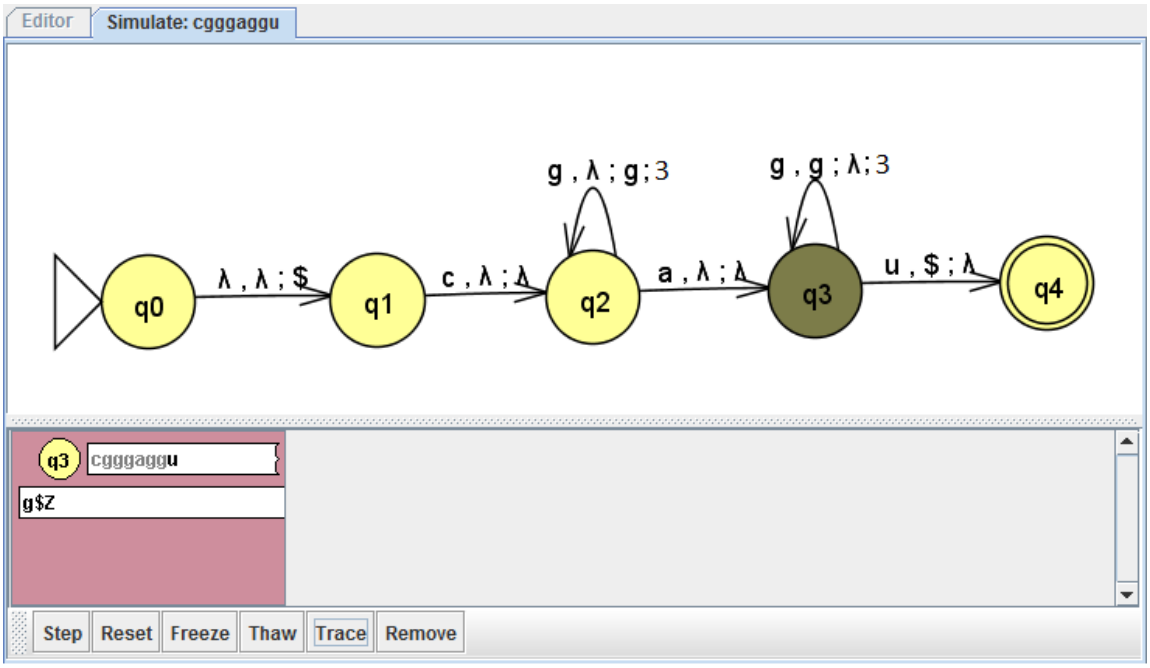




If MEPDA reaches the final state q4 with empty stack then the matching is successful with this input string.

Example 2: we use the same pattern from example 1 with a different input string: CGGGAGGU.

The simulation runs normally with the first steps, until the state q3 when MEPDA will loop 2 times and will try to loop for the last time according to the counter value, but instead of that the simulation will stop with error because the next symbol of the input string is not g.



Part IV

CONCLUSION

6

CONCLUSION

6.1 CONCLUSIONS

Discovering of motifs within biological sequences is one of the most crucial tasks in the field of computational biology, building the best tools to solve this problem helps biologists solve many issues related to biology.

The nature of the motif discovery task, which is the analysis of information from data sets represented in a text format, has made the task a target field of implementing and improving Patten matching algorithms and since pattern recognition algorithms have been very useful to the development of computational biology in many aspects in the last decades. We review in this thesis the basic information that need to be understood first for any computer scientist who wants to build algorithms that can help solve biological problems, we also provide many works that have been addressed in the context of motif discovery, showing their way of working, theirs strength and weakness.

Since pattern matching has been used very frequently within computational biology, we present in this thesis a novel algorithm in the context of pattern matching called MEPda, which can find a very specific kind of motifs called loop-motifs in a given biological sequence. The algorithm takes its power from using push-down automaton as a mechanism of matching process alongside with a counter to verify the accepted length of the loop in an optimistic way of looking.

In addition, our experimental results show high accuracy and much better runtime for finding and locating patterns that do or do not contain loops, by comparing it to many well-known algorithms in the context of pattern matching. As a conclusion the proposed algorithm MEPda shows an improvement of 40% in accuracy compared to Aho-Corasick and KMP algorithms, and a runtime efficiency of 60% better than exMotif and sMotif algorithms, also a runtime efficiency of 20% better than MoTeX algorithm.

As of now, MEPda is a word based algorithm. As future work, we aim to implement another extension of MEPda that can allow errors, instead of giving only one of two results, true or false, the next extension can give probability values.

As future works we have other ideas that can be transformed into pattern matching algorithms, we already started developing a new algorithm based on an extension of the Aho-Corasick (AC) algorithm, one of the most powerful and well known pattern matching algorithms that shows a significant progress in the field of pattern matching. The proposed extension is to adapt the AC algorithm to detect biological motif by overriding two of its basic functions, the first one is responsible of building a keyword tree from a set of patterns and applies a modified failure function in order to build relations between these keywords based on biological patterns roles. While the second function is responsible for the matching process, in this function we used a substitution matrix to obtain a probabilistic pattern matching score.

Part V

APPENDIX

7

WHAT WE HAVE ACQUIRED DURING THIS PERIOD

We have been very lucky during this period of Ph.D that allow us to learn more about many things: research, collaboration, people, area.

Groups

I was and still a member of the ISCB-SC (International Society of Computational Biology - Student Council) group, a group of young researchers (Ph.D. students, Post-Doc students, Professors, Professional workers) with a one goal which is sharing knowledge and informations about the field of bioinformatics.

During this membership we organize several research events includes international conferences, Skype meetings.. ext.

Conferences

1. **ISMB2014** (Organizer) International Society for Computational Biology (Boston, USA) *ISMB 2014 Proceedings Papers Committee. Bioinformatics 2014;30(12):i3-i8. doi:10.1093/bioinformatics/btu322.*
2. **ESCS2014** (Organizer) European Student Council Symposium (Strasbourg, French)
3. **ISIA2014** (Organizer + Participant) International Symposium on Informatics and Its Applications (M'sila, Algeria)
4. **ICSIP2013** (Participant) Third International Conference on Systems and Information Processing (Guelma, Algeria)
5. **ICACIS2012** (Participant) International Conference on Advanced Communication and Information Systems (Batna, Algeria)

6. **CITEE2012** (Participant) Internatonal Conference on Information Technology and Electrical Engineering (Yogyakarta , Indonesia)

Articles

1. Lounnas Bilal, Bouderah Brahim and Moussaoui Abdelouahab. *"Biological Motif Discovery Algorithm based on Mining Tree Structure"*. International Journal of Computer Applications 69(4):35-40, May 2013.

2. Lounnas Bilal, Bouderah Brahim and Moussaoui Abdelouahab. *"A novel algorithm for pattern matching based on modified push-down automata"*. Journal of Information Science and Engineering 32(2):409-430, March 2016.

BIBLIOGRAPHY

[1] (2011 (accessed October 10, 2014)). *STATISTICA Data Mining, Text Mining and Predictive Analytics Software, StatSoft*.

[2] (2013). Predict the trajectory of a handwritten signature. In *Icdar*. https://www.kaggle.com/c/icdar2013-stroke-recovery-from-offline-data.

[3] (2014(accessed February 27, 2015)). Molecular biology database/biological database/bioinformatics database - an overview. Technical report, Bioinformatics web. http://bioinformaticsweb.net/data.html.

[4] Aho, A. V. and Corasick, M. J. (1975). Efficient string matching: An aid to bibliographic search. *Commun. ACM*, 18(6):333–340.

[5] Alshahrani, A. and Khalil, M. (2013). Exact and like string matching algorithm for web and network security. In *Computer and Information Technology (WCCIT), 2013 World Congress on*, pages 1–4.

[6] Amal, D. A., Joe, C., and Raylene, Y. (2004). Basics of automata theory. Technical report, Stanford university.

[7] arthur lesk (2015 (accessed February 26, 2015)). Bioinformatics. Technical report, Encyclopedia Britannica Online.

[8] Beal, V. (2014 (accessed February 27, 2015)). Database. Technical report, webopedia encyclopedia.

[9] Bin Raies, A., Mansour, H., Incitti, R., and Bajic, V. B. (2013). Combining position weight matrices and document-term matrix for efficient extraction of associations of methylated genes and diseases from free text. *PLoS ONE*, 8(10):e77848.

[10] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[11] Brazma, A., Jonassen, I., Eidhammer, I., and Gilbert, D. (1998). Approaches to the automatic discovery of patterns in biosequences. *J. Comput. Biol.*, 5(2):279–305.

[12] Bucher, P. (1990). Weight matrix descriptions of four eukaryotic RNA polymerase II promoter elements derived from 502 unrelated promoter sequences. *J. Mol. Biol.*, 212(4):563–578.

[13] Cantone, D. and Faro, S. (2004). Searching for a substring with constant extra-space complexity. In Boldi, P. and Gargano, L., editors, *Fun with Algorithms*, Lecture Notes in Computer Science, pages 118–131. Springer Berlin Heidelberg.

[14] Cantone, D., Faro, S., and Giaquinta, E. (2010a). Bit-(parallelism)2: Getting to the next level of parallelism. In Boldi, P. and Gargano, L., editors, *Fun with Algorithms*, volume 6099 of *Lecture Notes in Computer Science*, pages 166–177. Springer Berlin Heidelberg.

[15] Cantone, D., Faro, S., and Giaquinta, E. (2010b). A compact representation of nondeterministic (suffix) automata for the bit-parallel approach. In Amir, A. and Parida, L., editors, *Combinatorial Pattern Matching*, volume 6129 of *Lecture Notes in Computer Science*, pages 288–298. Springer Berlin Heidelberg.

[16] Chao, K.-M. and Zhang, L. (2009). *Sequence comparison : theory and methods*. Computational Biology. Springer, London.

[17] Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., and Wirth, R. (2000). Crisp-dm 1.0 step-by-step data mining guide. Technical report, The CRISP-DM consortium.

[18] Chen, W. and Gao, Y. (2013). Face recognition using ensemble string matching. *Image Processing, IEEE Transactions on*, 22(12):4798–4808.

[19] Chen, X., Guo, L., Fan, Z., and Jiang, T. (2008). W-AlignACE: an improved Gibbs sampling algorithm based on more accurate position weight matrices learned from sequence and gene expression/ChIP-chip data. *Bioinformatics*, 24(9):1121–1128.

[20] Chin, F. Y. L. and Leung, H. C. M. (2005). Voting algorithms for discovering long motifs. Proceedings of Asia-Pacific Bioinformatics Conference.

[21] Chuanhan, L., Yongcheng, W., Derong, L., and Danglin, L. (2006). Two improved single pattern matching algorithms. In *Artificial Reality and Telexistence–Workshops, 2006. ICAT '06. 16th International Conference on*, pages 419–422.

[22] Comon, H., Dauchet, M., Gilleron, R., Löding, C., Jacquemard, F., Lugiez, D., Tison, S., and Tommasi, M. (2007). Tree automata techniques and applications. Available on: http://www.grappa.univ-lille3.fr/tata. release October, 12th 2007.

[23] Cox, R. (2007). Regular expression matching can be simple and fast (but is slow in java, perl, php, python, ruby, ...). Technical report.

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

[25] Deusdado, S. and Carvalho, P. (2009). Graspm: an efficient algorithm for exact pattern-matching in genomic sequences. *Int. J. Bioinformatics Res. Appl.*, 5(4):385–401.

[26] Dharmapurikar, S. and Lockwood, J. (2006). Fast and scalable pattern matching for network intrusion detection systems. *Selected Areas in Communications, IEEE Journal on*, 24(10):1781–1792.

[27] Diego, S. and de Ramon, A. (2008). Data mining web-tool prototype using monte carlo simulations. Master's thesis, School of Engineering, Blekinge Institute of Technology.

[28] diffen (2013 (accessed February 27, 2015)). Dna vs. rna. Technical report, Diffen encyclopedia.

[29] Dunham, M. (2003). *Data Mining Introductory and Advanced Topics*. An Alan R. Apt book. Prentice Hall/Pearson Education.

[30] Dusan, K. (2009). Formal pushdown automata formal definition and view. Technical report, Faculty of Information Technology.

[31] Edward, K. and Ajit, N. (2005). *Intelligent Bioinformatics: The Application of Artificial Intelligence Techniques to Bioinformatics Problems*. Wiley; 1 edition (May 27, 2005).

[32] Eskin, E. and Pevzner, P. A. (2002). Finding composite regulatory patterns in DNA sequences. *Bioinformatics*, 18 Suppl 1:S354–363.

[33] Fan, H., Yao, N., and Ma, H. (2009). Fast variants of the backward-oracle-marching algorithm. In *Internet Computing for Science and Engineering (ICICSE), 2009 Fourth International Conference on*, pages 56–59.

[34] Faro, S. and Lecroq, T. (2008). Efficient variants of the backward-oracle-matching algorithm. In Jan, H. and Jan, Ž., editors, *Proceedings of the Prague Stringology Conference 2008*, pages 146–160, Czech Technical University in Prague, Czech Republic.

[35] Faro, S. and Lecroq, T. (2013). The exact online string matching problem: A review of the most recent results. *ACM Comput. Surv.*, 45(2):13:1–13:42.

[36] Fayyad (1996 (accessed October 10, 2014)). *The Primary Tasks of Data Mining*.

[37] Ganesh, R., Siegele, D. A., and Ioerger, T. R. (2003). MOPAC: motif finding by preprocessing and agglomerative clustering from microarrays. *Pac Symp Biocomput*, pages 41–52.

[38] Ginsburg, S. (1966). *The Mathematical Theory of Context-Free Languages*. McGraw-Hill, Inc., New York, NY, USA.

[39] Graves, M. and Batchelor, B. (2003). *Machine Vision for the Inspection of Natural Products*. Springer.

[40] Griffiths-Jones, S., Bateman, A., Marshall, M., Khanna, A., and Eddy, S. R. (2003). Rfam: an RNA family database. *Nucleic Acids Res.*, 31(1):439–441.

[41] Guessarian, I. (1983). Pushdown tree automata. *Mathematical systems theory*, 16(1):237–263.

[42] Hagedoorn, M. (2000). *Pattern Matching Using Similarity Measures*. PhD thesis, University of utrecht.

[43] Heger, A., Lappe, M., and Holm, L. (2004). Accurate detection of very sparse sequence motifs. *J. Comput. Biol.*, 11(5):843–857.

[44] Holzinger, A., Stocker, C., Peischl, B., and Simonic, K.-M. (2012). On using entropy for enhancing handwriting preprocessing. *Entropy*, 14(11):2324–2350.

[45] Hon, L. S. and Jain, A. N. (2006). A deterministic motif finding algorithm with application to the human genome. *Bioinformatics*, 22(9):1047–1054.

[46] Hopcroft, J. E., Motwani, R., and Ullman, J. D. (2006). *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[47] Jiawei, H., Kamber, M., and Pei, J. (2012). *Data Mining: Concepts and Techniques (Third Edition)*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, Boston, third edition edition.

[48] Julia, P. and Michal, Z. (2009). Elaboration of a method for comparison of business intelligence systems which support data mining process. Master's thesis, Blekinge Tekniska university.

[49] Kahara, J. and Lahdesmaki, H. (2013). Evaluating a linear k-mer model for protein-dna interactions using high-throughput selex data. *BMC Bioinformatics*, 14(Suppl 10):S2.

[50] Kalsi, P., Peltola, H., and Tarhio, J. (2008). Comparison of exact string matching algorithms for biological sequences. In Elloumi, M., KÃŒng, J., Linial, M., Murphy, R., Schneider, K., and Toma, C., editors, *Bioinformatics Research and Development*, volume 13 of *Communications in Computer and Information Science*, pages 417–426. Springer Berlin Heidelberg.

[51] Kitano, H. (2002). Systems biology: A brief overview. *Science*, 295(5560):1662–1664.

[52] Kiwon, Y., Jean, H., Debaleena, C., Berg, T. L., and Samaras, D. (2012). Two-person interaction detection using body-pose features and multiple instance learning. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*. IEEE.

[53] Knuth, D. E., James H. Morris, J., and Pratt, V. R. (1977). Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2):323–350.

[54] Krasinski, T., Sakowski, S., and Poplawski, T. (2012). Autonomous pushdown automaton built on dna. *Informatica (Slovenia)*, 36(3):263–276.

[55] Külekci, M. O. (2009). Filter based fast matching of long patterns by using simd instructions. In Holub, J. and Žďárek, J., editors, *Proceedings of the Prague Stringology Conference 2009*, pages 118–128, Czech Technical University in Prague, Czech Republic.

[56] Leung, H. C. and Chin, F. Y. (2006). An efficient motif discovery algorithm with unknown motif length and number of binding sites. *Int J Data Min Bioinform*, 1(2):201–215.

[57] Levenshtein, V. (1965). Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information Transmission*, 1:8–17.

[58] Longtao, H., Binxing, F., and Jie, S. (2005). The wide window string matching algorithm. *Theoretical Computer Science*, 332(13):391 – 404.

[59] LoSacco, M. and Rodger, S. (1993). Flap: A tool for drawing and simulating automata. In *ED-MEDIA 93, World Conference on Educational Multimedia and Hypermedia*.

[60] Luscombe, N. M., Greenbaum, D., and Gerstein, M. (2001). What is bioinformatics? A proposed definition and overview of the field. *Methods Inf Med*, 40(4):346–358.

[61] Masters, H. V. (1927). A study of spelling errors,: A critical analysis of spelling errors occurring in words commonly used in writing and frequently misspelled,. *The University*.

[62] Mathura, V. and Kangueane, P. (2008). *Bioinformatics: A Concept-Based Introduction*. Springer Publishing Company, Incorporated, 1st edition.

[63] Melichar, B., Jan, H., and J, P. (2005). Text searching algorithms - volume 1: forward string matching. Technical report, Department of computer science and engineering, Czech technical university.

[64] Navarro, G. (2001). A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88.

[65] 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.

[66] Nesbit, J. C. (1986). The accuracy of approximate string matching algorithms. *J. Comput. Based Instruct.*, 13(3):80–83.

[67] Noble, D. (2002). Modeling the heart–from genes to cells to the whole organ. *Science*, 295(5560):1678–1682.

[68] Oded, M. and Lior, R. (2009). *Data Mining and Knowledge Discovery Handbook Second Edition*. Springer series in solid-state sciences (Book 14). Springer; 2nd ed. 2010 edition (October 1, 2010).

[69] Ono, H. and Ng, Y. K. (2005). Best fitting fixed-length substring patterns for a set of strings. In *Proceedings of the 11th Annual International Conference on Computing and Combinatorics*, COCOON'05, pages 240–250, Berlin, Heidelberg. Springer-Verlag.

[Patrice and Mounier] Patrice, B. and Mounier, H. Automated, transformation invariant, shape recognition through wavelet multiresolution.

[71] Pissis, S. (2014). Motex-ii: structured motif extraction from large-scale datasets. *BMC Bioinformatics*, 15(1):235.

[72] Qi, Y., Ye, P., and Bader, J. S. (2005). Genetic Interaction Motif Finding by expectation maximization–a novel statistical model for inferring gene modules from synthetic lethality. *BMC Bioinformatics*, 6:288.

[73] Rejeb, J. and Srinivasan, M. (2008). Extension of aho-corasick algorithm to detect injection attacks. In Sobh, T., editor, *Advances in Computer and Information Sciences and Engineering*, pages 207–212. Springer Netherlands.

[74] Remco, V. and Michiel, H. (1999). State-of-the-art in shape matching. Technical report, Principles of Visual Information Retrieval.

[75] Régnier, Mireille, D. A. (2004). Rare events and conditional events on random strings. *Discrete Mathematics and Theoretical Computer Science. DMTCS [electronic only]*, 6(2):191–213.

[76] Rouse, M. (2005 (accessed October 3, 2014)). *Text*. http://whatis.techtarget.com/definition/text.

[77] S-cool (2014 (accessed October 3, 2014)). *Pattern recognition*. http://www.s-cool.co.uk/a-level/psychology/attention/revise-it/pattern-recognition.

[78] Sandve, G. and Drablos, F. (2006). A survey of motif discovery methods in an integrated framework. *Biology Direct*, 1(1):11.

[79] Sargur, S., Chen, H., and Harish, S. (2008). On the discriminability of the handwriting of twins. *Journal of Forensic Sciences*, 53:430–446.

[80] Sellers, P. H. (1980). The theory and computation of evolutionary distances: Pattern recognition. *Journal of Algorithms*, 1(4):359 – 373.

[81] Serge, B., Jitendra, M., and Jan, P. (2001). Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:509–522.

[82] Sergios, T. and Konstantinos, K. (2009). *Pattern Recognition*. Academic Press, fourth edition.

[83] Shapira, D. and Daptardar, A. (2006). Adapting the knuth-morris-pratt algorithm for pattern matching in huffman encoded texts. *Inf. Process. Manage.*, 42(2):429–439.

[84] Shida, K. (2006). GibbsST: a Gibbs sampling method for motif discovery with enhanced resistance to local optima. *BMC Bioinformatics*, 7:486.

[85] Siddharthan, R., Siggia, E. D., and van Nimwegen, E. (2005). PhyloGibbs: a Gibbs sampling motif finder that incorporates phylogeny. *PLoS Comput. Biol.*, 1(7):e67.

[86] Sinha, S. (2003). Discriminative motifs. *J. Comput. Biol.*, 10(3-4):599–615.

[87] Thierry, L. (2007). Fast exact string matching algorithms. *Inf. Process. Lett.*, 102(6):229–235.

[88] Vilo, J., Brazma, A., Jonassen, I., Robinson, A., and Ukkonen, E. (2000). Mining for putative regulatory elements in the yeast genome using gene expression data. *Proc Int Conf Intell Syst Mol Biol*, 8:384–394.

[89] Vintsyuk, T. K. (1968). Speech discrimination by dynamic programming. *Cybernetics*, 4(1):52–57. Russian Kibernetika 4(1):81-88 (1968).

[90] Wagner, R. A. and Fischer, M. J. (1974). The string-to-string correction problem. *J. ACM*, 21(1):168–173.

[91] Wang, G., Yu, T., and Zhang, W. (2005). WordSpy: identifying transcription factor binding motifs by building a dictionary and learning a grammar. *Nucleic Acids Res.*, 33(Web Server issue):W412–416.

[92] Waterman, M. S. (1995). *Introduction to Computational Biology: Maps, Sequences and Genomes*. Chapman & Hall/CRC Interdisciplinary Statistics. Chapman and Hall/CRC; Softcover reprint of the original 1st ed. 1995 edition (January 1, 1995).

[93] Webopedia (2013 (accessed October 3, 2014)). *Handwriting recognition*. http://www.webopedia.com/TERM/H/handwriting_recognition.html.

[94] Wikipedia (2011 (accessed October 10, 2014)c). *RapidMiner*.

[95] Wikipedia (2014 (accessed October 10, 2014)a). *Educational data mining*.

[96] Wikipedia (2014 (accessed October 3, 2014)b). *Pattern recognition*. http://en.wikipedia.org/wiki/Pattern_recognition.

[97] Wolfe, K. (2013 (accessed October 3, 2014)). *The Use of Pattern Recognition*. http://www.ehow.com/about_5507022_use-pattern-recognition.html.

[98] Xindong, W. and Vipin, K. (2009). *The Top Ten Algorithms in Data Mining*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. Chapman and Hall/CRC; 1 edition (April 9, 2009).

[99] Xiong, J. (2006). *Essential Bioinformatics*. Cambridge University Press.

[100] Zhang, G., Zhu, E., Mao, L., and Yin, M. (2009). A bit-parallel exact string matching algorithm for small alphabet. In Deng, X., Hopcroft, J., and Xue, J., editors, *Frontiers in Algorithmics*, volume 5598 of *Lecture Notes in Computer Science*, pages 336–345. Springer Berlin Heidelberg.

[101] Zhang, Y. and Zaki, M. (2006a). Exmotif: efficient structured motif extraction. *Algorithms for Molecular Biology*, 1(1):21.

[102] Zhang, Y. and Zaki, M. (2006b). Smotif: efficient structured pattern and profile motif search. *Algorithms for Molecular Biology*, 1(1):22.