BABEŞ-BOLYAI UNIVERSITY
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Machine Learning Models for Solving Problems in Bioinformatics

PhD Thesis Abstract

PhD student: Maria Iuliana Bocicor
Scientific supervisor: Prof. Dr. Gabriela Czibula

September 2013

List of publications

Publications in ISI Web of Knowledge

Publications in ISI Science Citation Index Expanded

1. [CBC13] Gabriela Czibula, **Iuliana M. Bocicor** and Istvan Gergely Czibula. Temporal Ordering of Cancer Microarray Data through a Reinforcement Learning Based Approach. *PLoS ONE*, Vol. 8, No. 4, e60883, doi:10.1371/journal.pone.0060883, 2013. **(IF: 4.092)**
2. [CBC12] Gabriela Czibula, **Maria Iuliana Bocicor** and Istvan Gergely Czibula. Promoter Sequences Prediction Using Relational Association Rule Mining. *Evolutionary Bioinformatics*, Vol. 8, pages 181–196, 2012. **(IF: 1.229)**
3. [BCC11a] **Maria Iuliana Bocicor**, Gabriela Czibula and Istvan Gergely Czibula. A Distributed Q-Learning Approach to Fragment Assembly. *SIC Journal, Studies in Informatics and Control* Vol. 20, Issue 3, pages 221–232, 2011. **(IF: 0.671)**

Publications in ISI Conference Proceedings Citation Index

4. [BCC11b] **Maria Iuliana Bocicor**, Gabriela Czibula and Istvan Gergely Czibula. A Reinforcement Learning Approach for Solving the Fragment Assembly Problem. In *Proceedings of the 3th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC '11)*, pages 191-198, IEEE Computer Society, 2011.

Papers published in international journals and prooceedings of international conferences

1. [CCB13] Gabriela Czibula, Istvan Gergely Czibula and **Iuliana M. Bocicor**. A Comparison of Reinforcement Learning Based Models for the DNA Fragment Assembly Problem. *Studia Universitatis "Babes-Bolyai", Informatica*, LVIII(2), pages 90-102 2013. **(indexed Mathematical Reviews)**
2. [BCG$^+$13] **Iuliana Bocicor**, Giulio Caravagna, Alex Graudenzi, Claudia Cava, Giancarlo Mauri and Marco Antoniotti. Reconstructing Colorectal Cancer Progression through Copy Number Alteration Data. *Complex Systems Models in Biology and Medicine: Generic Properties and Applications* (Will be submitted).
3. [Boc12a] **Iuliana M. Bocicor.** A Study on using Reinforcement Learning for Temporal Ordering of Biological Samples. *Studia Universitatis "Babes-Bolyai", Informatica*, LVII(4), pages 63-74, 2012. **(indexed Mathematical Reviews)**
4. [BCG$^+$12] **Iuliana M. Bocicor**, Giulio Caravagna, Alex Graudenzi, Claudia Cava, Giancarlo Mauri and Marco Antoniotti. Ordering copy number alteration data to analyze colorectal cancer progression. Proceedings of NETTAB 2012: Workshop on Integrated Bio-Search, *EMBnet.journal*, Vol. 18, Suppl. B (NETTAB 2012), pages 84-86, 2012. **(indexed Google Scholar)**
5. [Boc12c] **Maria Iuliana Bocicor.** A Study on Using Association Rules for Predicting Promoter Sequences. *Studia Universitatis "Babes-Bolyai", Informatica*, LVII(2), pages 32-42, 2012. **(indexed Mathematical Reviews)**

6. [CBC11a] Gabriela Czibula, **Maria Iuliana Bocicor** and Istvan Gergely Czibula. A Distributed Reinforcement Learning Approach for Solving Optimization Problems. In *Proceedings of the 5th International Conference on Communications and Information Technology (CIT '11)* Greece, pages 25–30, 2011. **(indexed INSPEC)**

7. [CBC11c] Gabriela Czibula, **Maria Iuliana Bocicor** and Istvan Gergely Czibula. A Reinforcement Learning Model for Solving the Folding Problem. *IJCTA - International Journal of Computer Technology and Applications*, Vol. 2, Issue 1, pages 171–182, 2011. **(indexed Index Copernicus)**

8. [CBC11b] Gabriela Czibula, **Maria Iuliana Bocicor** and Istvan Gergely Czibula. An Experiment on Protein Structure Prediction using Reinforcement Learning. *Studia Universitatis "Babes-Bolyai", Informatica*, LVI(1), pages 25–34, 2011. **(indexed Mathematical Reviews)**

9. [CBC11d] Gabriela Czibula, **Maria Iuliana Bocicor** and Istvan Gergely Czibula. Solving the Protein Folding Problem Using a Distributed Q-Learning Approach. *International Journal of Computers*, Volume 5, Issue 3, pages 404–413, 2011. **(indexed INSPEC)**

10. [CCB11a] Istvan Gergely Czibula, Gabriela Czibula and **Maria Iuliana Bocicor**. A Software Framework for Solving Combinatorial Optimization Tasks. *Studia Universitatis "Babes-Bolyai", Informatica, Special Issue*, LVI(3), pages 3–8, 2011. **(indexed Mathematical Reviews)**

Papers published in proceedings of national conferences

1. [Boc12b] **Maria Iuliana Bocicor**. Experiments on Promoter Sequences Prediction using Association Rules. In *Proceedings "Zilele Academice Clujene 2012, Departamentul de Informatica"*, Presa Universitara Clujeana, pages 32–35, Cluj Napoca, 2012.

2. [CCB11b] Istvan Gergely Czibula, Gabriela Czibula and **Maria Iuliana Bocicor**. A Reinforcement Learning Based Framework for Solving Optimization Problems. In *Post proceedings of Knowledge Egineering Principles and Techniques*, pages 235–246, Presa Universitara Clujeana, 2011.

Other publications

1. [Boc11a] **Maria Iuliana Bocicor**. Invatarea automata pentru identificarea regiunilor promotor in ADN. *National Symposum "Interferente", 3rd Edition*, Universitatea de Nord Baia Mare, ISBN 978-606-536-226-0, pages 68–70, 2012. **(CNCSIS)**

2. [Boc11b] **Maria Iuliana Bocicor**. Modele pentru problema plierii proteinei. *National Symposum "Interferente", 2nd Edition*, Universitatea de Nord Baia Mare, ISBN 978-606-536-146-1, pages 191–193, 2011. **(CNCSIS)**

3. [Boc10a] **Maria Iuliana Bocicor**. Algoritmi evolutivi aplicati in chemoterapie. *National Symposum "Interferente", 1st Edition*, Ceconi, Baia-Mare, ISBN 978-606-8086-29-3, pages 145–148, 2010.

4. [Boc10b] **Maria Iuliana Bocicor**. Bioinformatica si aplicatiile ei. *Scoala Maramureseana*, Nr. 41-45, ISSN 1583-2171, pages 222–223, 2010.

In the case of co-authored publications, all authors have equally contributed to: conceiving, designing and performing the experiments; analyzing the data; developing the structure and arguments of the papers; writing the manuscripts; making critical revisions and approving the final versions of the papers.

Contents

Introduction **4**

1 Bioinformatics Problems. Background **7**

1.1 Fundamental Concepts of Molecular Biology 7

1.2 Computational Intelligence and Machine Learning in Bioinformatics. Challenges and Perspectives 8

1.3 Promoter Sequences Prediction 8

1.4 The DNA Fragment Assembly Problem 9

1.5 Protein Tertiary Structure Prediction 9

1.6 The Biological Temporal Ordering Problem 10

2 Novel Relational Association Rules based Mining Approach for Promoter Sequences Prediction **11**

2.1 Relational Association Rules. Background 11

2.2 A New Method For Promoter Sequences Prediction 12

2.3 Experimental Evaluation 14

2.4 Conclusions and Further Work 16

3 New Reinforcement Learning Based Approaches in Bioinformatics **17**

3.1 Reinforcement Learning. Background 17

3.2 Proposed Reinforcement Learning Based Models. Theoretical Considerations 19

3.3 Reinforcement Learning Based Model for the DNA Fragment Assembly Problem 21

3.4 Reinforcement Learning Based Model for the Protein Tertiary Structure Prediction Problem 23

3.5 Conclusions and Further Work 25

4 New Approaches to the Biological Temporal Ordering Problem **26**

4.1 Temporal Ordering of Colorectal Cancer Samples through Copy Number Alteration Data 26

4.2 Reinforcement Learning Based Model for the Biological Temporal Ordering Problem 29

4.3 A Reinforcement Learning based Software Framework 32

4.4 Conclusions and Further Work 33

Conclusions **34**

Introduction

This Ph.D. thesis is the result of my research in the field of machine learning, specifically focusing on *machine learning models for solving problems in bioinformatics.* This research was started in 2010, under the supervision of Prof. Dr. Gabriela Czibula.

The primary research direction we are focusing on is applying machine learning models to solve complex problems in the field of bioinformatics.

Machine Learning (ML) [Mit97], a branch of artificial intelligence, is concerned with the development of algorithms that enable computers to learn and improve automatically through experience. ML is an interdisciplinary field which uses knowledge and results obtained in a high variety of fields, like artificial intelligence, mathematics, probability and statistics, information theory, psychology, neurobiology and others. Even though computers are yet not capable to learn as well as human beings, a multitude of theoretical methods and algorithms that improve from experience have been developed, which are very effective for a wide range of complex problems.

Bioinformatics is an object of interdisciplinary research and it tries to solve problems from fields like biology, biochemistry or medicine by applying computational techniques for the collection, management, organization and especially for the analysis of biological information. Some of the most important problems in bioinformatics are too complex to be solved from first principles. For such difficult tasks, ML methods have proven to be very well suited.

The particular problems we have decided to approach are among the greatest challenges in bioinformatics and various computational intelligence algorithms were designed to find good solutions to these problems. Our main goal is to find new ML models and algorithms that could offer solutions that are comparable and even better than the existing ones, in terms of solution quality as well as of computational time.

Therefore, our research is mainly focused on two directions. The first one is the application of *relational association rules learning* [SCC06] to solve classification problems in bioinformatics. The second direction refers to applying *reinforcement learning* based techniques in order to solve NP-complete combinatorial optimization problems in bioinformatics.

In addition to the above mentioned primary research directions, we present in this thesis a novel methodology targeting a specific problem in bioinformatics (the temporal ordering problem) and a particular type of biological data. This approach was developed in collaboration with the BIMIB research group from University Milano-Bicocca, Italy. Finally, this thesis also presents a new programming interface for solving optimization problems using reinforcement learning techniques.

The thesis is structured in four chapters, as follows.

The first chapter, **Bioinformatics Problems. Background**, shortly describes the field of bioinformatics and presents some of its most challenging problems. We begin by providing a brief introduction to molecular biology. Further, we present how some fundamental modern machine learning and computational intelligence techniques and algorithms have been successfully applied to various problems in bioinformatics. The four major problems approached in this thesis are described in greater detail: *the promoter sequences prediction*, *the DNA (Deoxyribonucleic Acid) fragment assembly*, *the protein tertiary structure prediction* and *the biological temporal ordering problem.*

Chapters 2, 3 and 4 contain our original contributions that have been carried out towards proposing new ML based models for solving complex problems in bioinformatics.

Chapter 2, **Novel Relational Association Rules based Mining Approach for Promoter Sequences Prediction**, presents a new classification model based on relational association rules mining, that we propose for the identification of promoter sequences in the DNA. We begin by offering a description of relational association rules, along with an algorithm for identifying the relevant ordinal association rules [CSTM06]. We then introduce our supervised learning technique for promoter sequences recognition, based on relational association rules mining. We describe the classification algorithm and two of its extensions. All three algorithms are experimentally evaluated and the obtained results are analyzed. The main classifier is compared with other classifiers from the literature and we also offer comparisons between this algorithm and its other two extensions. The conclusions of the chapter and possible further work are outlined in its last section.

Chapter 3, **New Reinforcement Learning Based Approaches in Bioinformatics**, introduces two new reinforcement learning based models and a distributed reinforcement learning based approach, used to offer solutions to two important problems in bioinformatics. We begin by presenting basic theoretical notions about reinforcement learning and continue to introduce three new general reinforcement learning based models for a certain class of combinatorial optimization problems. We also present a new intelligent action selection mechanism to be used in the reinforcement learning process. These models are particularized, applied and experimentally evaluated on two bioinformatics problems: DNA fragment assembly and protein tertiary structure prediction. The proposed approaches are analysed and compared one against the other, as well as with other approaches existing in the literature. Finally, the chapter contains its conclusions and the research directions that will be further investigated.

Chapter 4, **New Approaches to the Biological Temporal Ordering Problem**, presents two different approaches that we proposed for the biological temporal ordering problem. Furthermore, this chapter also introduces a new programming interface for solving optimization problems using reinforcement learning techniques, which is applied to find solutions to the temporal ordering problem. The first approach, which was developed in collaboration with a research team during my research internship at the University of Milano-Bicocca, introduces a new methodology developed for the temporal ordering problem. An experimental evaluation of the algorithm implementing this methodology on a colorectal cancer case study is presented and the results are analyzed. Then, we show how one of the reinforcement learning based models introduced in Chapter 3 is adapted and modified to approach the biological temporal ordering problem. The approach is experimentally evaluated on several real life gene expression data sets. The obtained results are analyzed and compared with other approaches existing in the literature. Several variations to this original approach are proposed and the results are compared against each other. The last subchapter introduces a new reinforcement learning based software framework and shows how this can be used to develop an application for the temporal ordering problem. In the end, we present the conclusions of the chapter, as well as other research directions that will be further investigated.

The original contributions introduced in this thesis are contained in Chapters 2, 3 and 4 and they are as follows:

- A supervised learning model for predicting promoter sequences, based on relational association rules mining - *PCRAR* (Subchapter 2.2) [CBC12].
- Supervised learning algorithms for predicting promoter sequences, based on relational association rules mining (Section 2.2.4 and Subsections 2.2.4.2, 2.2.4.3) [CBC12, Boc12c, Boc12b].
- Experimental evaluations of the algorithms on a case study, a comparison of the proposed approach with similar existing ones (Section 2.3.2) [CBC12] and comparisons of the algorithms (Subsection 2.3.2.2) [Boc12c, Boc12b].
- Two general reinforcement learning based models, the *path finding model* and the *permutation model* and a *distributed reinforcement learning based approach* for a specific type of combinatorial optimization problems (Sections 3.2.1, 3.2.2) [CBC13, CCB13, Boc12a, BCC11a, BCC11b, CBC11c, CBC11b].
- A new intelligent action selection policy defined so as to better guide the reinforcement learning agent towards good solutions (Section 3.2.4) [CBC13].
- Particularization of the three models for the DNA fragment assembly problem, as well as experimental evaluations on several DNA sequences, analysis of the results and comparisons with other approaches from the literature (Sections 3.3.1, 3.3.2, 3.3.4) [BCC11b, BCC11a].
- A comparison between the path finding model and the permutation model, applied on a small DNA sequence and analysis of the results (Sections 3.3.3, 3.3.4) [CCB13].
- A particularization of the path finding model and of the distributed approach for the protein tertiary structure prediction problem, as well as experimental evaluations on several small protein sequences, analysis of the results and comparisons with other approaches from the literature (Subchapter 3.4) [CBC11c, CBC11b, CBC11d, CBC11a].

- A new methodology which adapts a solution that was previously proposed for gene expression data [GBJ08] to chromosomal copy number alterations (Section 4.1.2) [BCG+12, BCG+13].
- Experimental evaluations of the newly introduced methodology on a set of static copy number alteration data taken from patients at different stages of colorectal cancer and comparisons of the results (Section 4.1.3) [BCG+12].
- A particularization of the path finding model for the biological temporal ordering problem (Section 4.2.1) [CBC13] and experimental evaluations on several real-life gene expression data sets, a new evaluation measure to quantify the performance of our algorithm (Section 4.2.2), as well as analysis of the results and comparisons with other approaches from the literature (Section 4.2.4) [CBC13].
- Q-Learning based algorithms implementing the path finding model for solving the temporal ordering problem (Section 4.2.3) [CBC13, Boc12a] and experimental evaluations of all these algorithms on a real-life gene expression yeast data set, analyses and comparisons of the obtained results (Sections 4.2.3, 4.2.4) [Boc12a].
- A generic, reinforcement learning based software framework and a particular application developed using this framework for the temporal ordering problem (Subchapter 4.3) [CCB11a, CCB11b].

Chapter 1

Bioinformatics Problems. Background

Bioinformatics is an object of interdisciplinary research and it tries to solve problems from fields like biology, biochemistry or medicine using methods from mathematics, statistics and computer science [HMTA08]. It is the application of computational techniques for the management, organization and especially for the analysis of biological information. Bioinformatics is an interface between computational sciences and biology. Its major goal is to process the huge amounts of information in order to elucidate the functioning of living organisms.

1.1 Fundamental Concepts of Molecular Biology

In order to understand the subject of bioinformatics we present the basic elements that are studied in molecular biology and which are used in bioinformatics problems. The concepts described in the following are taken from the work of Brazma *et al.* [BPSS01].

The *genome* of all living organisms is encoded in the DNA (Deoxyribonucleic Acid) and it represents the totality of their hereditary information. DNA may be single of double stranded. A strand of DNA is a chain composed of four types of complex organic molecules, called *nucleotides* - adenine (A), guanine (G), cytosine (C) and thymine (T). Here is an example of such a DNA strand: *AGTCCAAGCTT*. When two DNA chains are linked together, they form a stable structure which is known as the DNA double helix. RNA, or ribonucleic acid, is composed of a single chain formed of 4 types of nucleotides: adenine, guanine, cytosine and uracil (that replaces the thymine from the DNA).

Genes are segments of the DNA material, being considered the basic molecules that carry the hereditary information. They code for specific proteins. A gene may or may not be expressed in a certain cell, meaning that it leads or not to the synthesis of a gene product (this can be a protein or RNA). As all cells in an organism contain the same genetic information (the DNA is identical), the differences in gene expression are responsible for cell differentiation.

Amino acids are small molecules. They can be integrated into the large molecules (macromolecules), or may have independent roles. Amino acids link in a specific order to form a protein, therefore they may also be considered the building blocks of proteins. There are 22 amino acids that compose proteins of the human body, being denoted using the letters of the alphabet.

Proteins are very important molecules, composed of sequences of amino acids that can link in any order. The amino acids sequence forms the primary structure of the protein, which can be represented as a string of symbols representing the 22 amino acids (a protein can be seen as a word over the alphabet that is formed of the 22 letters representing the amino acids). As soon as it is synthesized as a linear sequence of amino acids, a protein folds in a matter of seconds to a stable three dimensional structure called the protein's native state. It is assumed that the information for the folding process is contained exclusively in the sequence of amino acids. The secondary structure of a protein is the general three dimensional form of local segments. It consists of local interactions between the amino acids. As a result of these interactions, certain parts of the protein chain come in contact with each other and because of the forces of attraction and repulsion the molecule adopts a fixed and relatively stable three dimensional structure - the tertiary structure. This structure of the protein is very important, as it defines the protein's function.

1.2 Computational Intelligence and Machine Learning in Bioinformatics. Challenges and Perspectives

Some of the most important problems in bioinformatics are too complex to be solved from first principles. The huge amounts of data, the fact that significant parts of it are unlabeled or that the data contains much noise are obstacles that make it impossible to solve these problems with traditional methods and algorithms. For these reasons, computational intelligence (CI) and machine learning (ML) methods seem to be very well suited for such tasks [PPN09]. These methods also have a certain degree of flexibility regarding the data inputs, and have the possibility to progressively expand in order to meet the requirements of rapidly accumulating data resulted from biology research.

There is a great number of examples that illustrate how CI and ML techniques can be applied to solve bioinformatics problems.

Gene expression refers to the process by which the information from a gene is converted into functional gene products (RNA or proteins). Modern microarray technology is used experimentally to detect the levels of expression of thousands of genes, across different conditions and over time. After gathering the data, one of the first steps in analyzing it is clustering. Various CI and ML related algorithms have been proposed to cluster gene expression data: an associative clustering neural network (ACNN) [YLGW02], a hybrid clustering approach based on particle swarm optimisation (PSO) and self organizing maps (SOM) [XDE+03], a fuzzy k-means method [AHO03] or k-nearest neighbour techniques [NMB+03, OP09]. Another type of analysis of gene expression data refers to selecting differentially expressed genes. For *gene selection* the literature offers different methods: probabilistic neural networks with a feature selection method [HL03], evolutionary computation coupled with artificial neural networks (ANNs) [Fog05] or bayesian networks, radial basis function networks and neural trees [HCP+00].

The *secondary structure* of proteins or RNA is the general three dimensional form of local segments, obtained due to interactions between the smaller molecules. The problem of determining this structure can be mapped to a standard classification problem. Several classification approaches that predict the secondary structure exist in the literature: methods that use ANNs [NT88, Ste93], k-nearest neighbor and fuzzy k-nearest neighbor [GP08, BS06], support vector machines (SVMs) [LHJYFHB04] and hidden Markov models (HMMs) [MGR06, YV04].

Protein sequence classification refers to characterizing new proteins based on their sequences and detecting close evolutionary relationships among sequences. Generative and discriminative methods have been proposed in the literature to solve this classification problem: a generative HMM for a protein family [JDH99], a SVM technique [LEN02] or a methodology for constructing a neural network classifier [WLD03].

Sequence alignment plays an important role in molecular sequence analysis. It refers to the process of arranging the primary sequences of DNA, RNA or protein to identify regions of similarity that may be a consequence of functional, structural or evolutionary relationships between the sequences. For this problem, several methods have been proposed: genetic algorithms [HYYY02, CL05], an ant colony optimization (ACO) method [CPCC06], HMMs [RK03] or a fuzzy logic method [NVNM07].

Other major problems in bioinformatics that will be approached in this thesis using different machine learning models that we introduced will be presented in greater detail in the following subchapters. These problems are: *promoter sequences prediction*, *DNA fragment assembly*, *protein tertiary structure prediction* and the *biological temporal ordering problem.*

1.3 Promoter Sequences Prediction

Promoters are regions of the DNA sequence that signal the start of a gene during the process of transcription, a first process involved in protein synthesis. The *problem of promoter identification* is of major importance within bioinformatics, for two main reasons. First, identifying promoters is a significant step in the process of detecting genes. Second, promoters are essential in the regulation of the expression of genes. As the conditions for a DNA sequence to function as a promoter are not known, machine learning methods are suitable to approach this problem because they can learn useful descriptions of concepts when given only instances - DNA sequences that are assumed to contain underlying but unknown patterns of base pairs [TYA08].

In the context of supervised machine learning, the identification of promoters can be stated as follows [CS94]: given two sets of DNA sequences of fixed length, one containing sequences with known promoter regions and the other one containing sequences without the presence of this signal, generate a classifier able to predict whether a fixed length "window" of a DNA sequence contains or not promoter regions.

Several machine learning approaches have been applied in order to recognize biological sequences (such as promoters) that enable the transcription process: a hybrid learning system called KBANN (Knowledge-Based Artificial Neural Networks), that combines explanation based learning and empirical learning [TSN90], other neural network approaches [PE95], a grey relational analysis method [TPAG11], SVM approaches [KP04]. To our knowledge, association rule mining has not been used in the literature for the specific task of identifying promoters.

1.4 The DNA Fragment Assembly Problem

Determining the order of nucleotide bases, or the process of DNA sequencing, has nowadays become of great importance in basic biology research, as well as in medicine, biotechnology or forensic biology. In order to sequence large strands of DNA, they are first broken into smaller pieces, which can be directly sequenced. The *fragment assembly problem* consists in reconstructing the original molecule's sequence from the smaller fragment sequences [LK04], based on the fragments' overlaps. This problem is known to be NP-complete [Pev00], therefore accurate solutions are very difficult to obtain. Moreover, different sequencing errors that affect the fragments raise new obstacles in solving the problem.

Next, we will illustrate the assembly process, by using a simple example, taken from [Kos07]. Let us assume that, for the DNA sequence $TTACCGTGC$, we are given the set of fragments: $F_1 = ACCGT$, $F_2 = CGTGC$, $F_3 = TTAC$ and $F_4 = TACCGT$. First we need to determine the overlap of each fragment with the other three fragments. This is usually done using an alignment algorithm and a similarity measure. Then, we need to find the order of these fragments, based on the computed similarities. The order is: $F_3F_4F_1F_2$.

$$\begin{array}{cccccccccccc}
F_3 & \rightarrow & T & T & A & C & - & - & - & - & - \\
F_4 & \rightarrow & - & T & A & C & C & G & T & - & - \\
F_1 & \rightarrow & - & - & A & C & C & G & T & - & - \\
F_2 & \rightarrow & - & - & - & - & C & G & T & G & C \\
\hline
 & & T & T & A & C & C & G & T & G & C
\end{array}$$

Various heuristic methods have been developed in the literature to approach the DNA fragment assembly problem: (improved) genetic algorithms [KC06, PFB95, LATK06], clustering heuristic algorithms [LK04], ACO algorithms [MC03, WCT06] and also supervised learning approaches, like recurrent neural networks [AAdFG99].

1.5 Protein Tertiary Structure Prediction

Protein structure prediction is one of the the greatest challenges of bioinformatics, being an important research direction due to its numerous applications in medicine (drug design, disease prediction) and genetic engineering (cell modelling, modification and improvement of the functions of certain proteins). As soon as it is synthesized as a linear sequence of amino acids, a protein folds, in a matter of seconds, to a stable three-dimensional structure, called the protein's native state, which determines the protein's function.

An important class of abstract models for proteins are lattice-based models - composed of a lattice that describes the possible positions of amino acids in space and an energy function of the protein, that depends on these positions. The goal is to find this function's global minimum, as it is assumed that a protein in its native state has a minimum free energy and the process of folding is the minimization of this energy [Anf73]. One of the most popular bidimensional lattice-models is Dill's Hydrophobic-Polar

(HP) model [Dil85], which is based on the observation that the hydrophobic forces are very important factors in the protein folding process, guiding the protein to its native three dimensional structure. In this model, each amino acid is regarded as either hydrophobic (repelled by water) or polar (having an affinity for water) and the energy function is defined so as to capture the interactions between neighboring hydrophobic amino acids.

The protein tertiary structure prediction in the HP model has been shown to be NP-complete [BL98], therefore various approximation and heuristics methods approach this problem. Among these, we remark ACO algorithms [SH05, TMM08], genetic algorithms [UM93], hybrid algorithms - that combine genetic algorithms and tabu search [ZWL$^+$09], evolutionary models with hill-climbing genetic operators [Chi10] and also supervised machine learning techniques, like SVMs and ANNs [DD01].

1.6 The Biological Temporal Ordering Problem

Temporal modeling and analysis and more specifically, temporal ordering are very important problems within the fields of bioinformatics and computational biology, as the temporal analysis of the events characterizing a certain biological process could provide significant insights into its development and progression. In many situations, ordering a given data set of instances in time provides more significant information than assigning them to certain classes. Therefore, from a machine learning perspective, the general problem of temporal ordering is comparable, as importance, to the classification problem [CSS99].

The temporal ordering problem can be expressed in various forms. One definition of this problem refers to determining and describing the sequence of events that characterize a biological process. If the process in question is cancer, for instance, the goal is to find a temporal order for the genetic and pathway alterations that occur during the genesis and evolution of this disease. It is known that most tumors develop as a result of mutations that appear in certain key genes (oncogenes or tumor suppressor genes) [Fra07]. Therefore, studying the order in which these mutations happen could lead to a better understanding of the evolution of cancer. Several works exist in the literature that approach the temporal ordering problem as described above. Some of these methods are probabilistic [HHL06], based on bayesian networks [GBHB09, GEL$^+$11], others focus on building tree models of possible genetic events [DJK$^+$99, DJK$^+$00, BJK$^+$05a, BJK$^+$05b, PSBM09] or on constructing mathematical models [ACB$^+$10, CBL$^+$12] to identify the order of gene mutations on cancer development.

A second research direction focuses on a different form of the temporal ordering problem, which refers to constructing a sorted collection of multi-dimensional biological data that reflects an accurate temporal evolution of a biological process. There are mainly two works in the literature that approach the problem as formulated here, both of them using gene expression data. The first one [GBJ08] uses the travelling salesman problem in order to retrieve a correct ordering of gene expression samples, while the second [MLK03] is based on minimum spanning trees and PQ-trees.

Chapter 2

Novel Relational Association Rules based Mining Approach for Promoter Sequences Prediction

In this chapter we are approaching the problem of promoter sequences prediction and we are proposing a classification model based on relational association rules (RARs) mining for the identification of promoter sequences in the DNA.

The approaches presented in this chapter are original works published in [CBC12, Boc12c, Boc12b].

Relational association rules [SCC06] were introduced in order to be able to capture various kinds of relationships between record attributes. Based on the idea of discovering RARs within a data set, we propose a classification model for the problem of promoter sequences prediction. We have started from the intuition that in the problem of deciding if a DNA sequence contains or not promoter regions, relationships between the nucleotides that form the DNA sequence may be relevant. These relationships could express quantitative information that may exist in a DNA sequence and it is likely that this type of relationships could significantly influence the classification task. We are focusing on developing a machine learning based computational model that will be powerful enough to capture aspects that are relevant in distinguishing between DNA sequences that contain or not a promoter region. In building our classifier, we try to interpret DNA sequences both by their biological and chemical properties and to exploit the benefit of data mining techniques to uncover hidden patterns in data.

2.1 Relational Association Rules. Background

Association rule learning is a popular method for discovering interesting relations or correlations that exist between data in large data sets. Given a set of attributes, called items, and a set of transactions, composing the input data set, where each transaction contains a subset of items, an association rule is an implication of two item sets. There are several types of association rules that can be discovered in data (binary, quantitative, fuzzy), one of these being ordinal association rules, which were introduced in [MML01]. However, there are real world situations in which ordinal association rules are not powerful enough to describe data regularities. Therefore, the definition of ordinal association rules was extended in [SCC06] towards relational association rules, in order to be able to capture various kinds of relationships between record attributes.

As in [SCC06], let $R = \{r_1, r_2, \ldots, r_n\}$ be a set of *instances* (entities or records in the relational model), where each instance is characterized by a list of m attributes, $(a_1, \ldots, a_m)$. We denote by $\Phi(r_j, a_i)$ the value of attribute a_i for the instance r_j, $i \in \{1, \ldots, m\}$, $j \in \{1, \ldots, n\}$. Each attribute a_i takes values from a domain D_i, which contains the empty value. We denote by M the set of all possible relations that can be defined on $D_i \times D_j$. In [SCC06], a *relational association rule* is defined as an expression $(a_{i_1}, a_{i_2}, a_{i_3}, \ldots, a_{i_\ell}) \Rightarrow (a_{i_1}\ \mu_1\ a_{i_2}\ \mu_2\ a_{i_3} \ldots \mu_{\ell-1}\ a_{i_\ell})$, where $\{a_{i_1}, a_{i_2}, a_{i_3}, \ldots, a_{i_\ell}\} \subseteq \mathcal{A} = \{a_1, \ldots, a_m\}$, $a_{i_j} \neq a_{i_k}$, $j, k = 1..\ell$, $j \neq k$ and $\mu_i \in M$ is a relation over $D_{i_j} \times D_{i_{j+1}}$, D_{i_j} being the domain of the attribute a_{i_j}. The *support* of a rule is the percent of cases (out of the n instances) when $a_{i_1}, a_{i_2}, a_{i_3}, \ldots, a_{i_\ell}$ occur together. As in [SCC06], we denote by $R' \subseteq R$ the set of instances where $a_{i_1}, a_{i_2}, a_{i_3}, \ldots, a_{i_\ell}$ occur together and $\Phi(r_j, a_{i_1})\ \mu_1\ \Phi(r_j, a_{i_2})\ \mu_2\ \Phi(r_j, a_{i_3})\ \ldots\ \mu_{\ell-1}\ \Phi(r_j, a_{i_\ell})$

is true for each instance r_j from R'. Then $c = |R'|/|R|$ is the *confidence* of the rule.

The *length* of a relational association rule is the number of attributes in the rule. This length can be at most equal to the number m of the attributes describing the data.

The users usually need to uncover interesting RARs that hold in a data set, i. e. relational rules which hold in a minimum number of instances, having the support at least s_{min}, and the confidence at least c_{min} (s_{min} and c_{min} are user-provided thresholds).

2.1.1 The *DOAR* algorithm

An A-Priori [AS94] like algorithm, called *DOAR - Discovery of Ordinal Association Rules*, for identifying the relevant ordinal association rules that hold within a data set was introduced in [CSTM06]. It efficiently finds all ordinal association rules (i.e. RARs in which the relations are ordinal) of any length, that hold over a data set.

The *DOAR* algorithm identifies ordinal association rules using an iterative process that consists in length-level generation of candidate rules, followed by the verification of the candidates for minimum support and confidence compliance. *DOAR* performs multiple passes over the data set. In the first pass, it calculates the support and confidence of the 2-length rules and determines which of them are interesting. Then, each subsequent pass starts with a seed set of $(k-1)$-length $(k \geq 3)$ interesting rules, found in the previous pass, which is used to generate new possible k-length interesting rules, called candidate rules. The candidate generation process is a key element of the *DOAR* algorithm. A scan over the data is performed in order to compute the actual support and confidence of the candidate rules. At the end of this step, the algorithm keeps the rules that are deemed interesting, which will be used in the next iteration. The process stops when no new interesting rules were found in the latest iteration. More about the *DOAR* algorithm and its theoretical validation is given in [CSTM06].

The *DOAR* algorithm is extended in our approach towards the *DRAR* algorithm (*Discovery of Relational Association Rules*) for finding interesting RARs, i.e. association rules which are able to capture various kinds of relationships between record attributes. Our current implementation is able to find all interesting RARs of any length, as well as all maximal interesting relational association rules of any length, i.e. if an interesting rule of a certain length can be extended with one attribute and it remains interesting, only the extended rule is kept.

2.2 A New Method For Promoter Sequences Prediction

In this subchapter we propose a supervised learning technique in order to predict promoter sequences, based on relational association rules mining, called *PCRAR* (*Promoter sequences Classifier using Relational Association Rules*). The classifier was built with the purpose of distinguishing DNA sequences that contain promoters from those that do not. Our *PCRAR* classifier is not based on any particular biological mechanisms, its strength consisting in its ability to automatically learn the differences between DNA sequences that include or not promoter regions, when given as input only these sequences and no other extra biological information.

The problem that we are focusing on is a binary classification problem. There are two possible classes: one contains the *positive instances*, or the DNA sequences containing promoters and the other one contains the *negative instances*, or the non-promoters.

The main idea of our approach is the following. In a supervised learning scenario for predicting promoter sequences, two sets containing positive and negative instances are given. These sets will be used for training the classifier. During training, the *DRAR* algorithm will be used. Even if this algorithm can be used to discover all the relational rules, of any length in a data set, at first we used it to discover only the binary RARs, i.e. rules of length two. We detect in the training data sets all the interesting binary relational rules (rules between two attributes), with respect to the user-provided support and confidence thresholds). After the training is completed, when a new instance (DNA sequence) has to be classified (as positive or negative), we reason as follows. Considering the binary rules discovered during training in the set of *positive* and *negative* instances, we compute a score for each instance, defined so as to range between 0 and 1 and which denotes the degree to which an instance can be considered positive. If this score is greater than or equal to 0.5, then the query instance will be classified as a *positive* instance, otherwise it will be classified as a *negative* instance.

We consider a DNA sequence (instance) as an n-dimensional chain $S = (s_1, s_2, \ldots, s_n)$ containing the four letters A, T, G and C, which represent the nucleotides composing the DNA (see Subchapter 1.1). Consequently, the attribute (feature) set characterizing the instances is an n-dimensional list where the i-th attribute corresponds to the i-th nucleotide from the DNA sequence, $i \in \{1, \ldots, n\}$. Therefore, each attribute has 4 possible values: the characters A, T, G and C.

For classifying a DNA sequence as containing or not a promoter region, the following steps will be performed:

1. Relations definition.
2. Data pre-processing.
3. Training/building the classifier.
4. Testing/classification.

2.2.1 Relations Definition

This step deals with defining the relations between the attributes' values that will be used in the RAR mining process. More exactly, we are focusing on identifying relations between two nucleotides from a DNA sequence (A, T, G or C), relations that would be relevant for deciding if the sequence contains or not a promoter region.

A first stage refers to searching for several computed chemical and physical properties that may characterize each nucleotide. We extracted from PubChem [BWTB08] 5 measurable properties: molar mass, density, topological polar surface area, heavy atom count and complexity. To these, we added base composition, one of the most fundamental features of a DNA sequence which refers to the percentages of each of the four different nucleotides, on one strand of DNA. Consequently, we associate to each nucleotide (attribute value) a list of six numerical codes, representing the values of the six above enumerated properties.

The second stage consists in identifying which of the six types of codes associated to the attributes would be relevant for the classification task. In this direction, a statistical analysis is performed on the training data sets to determine those codes that provide attributes highly correlated with the target output. To determine the dependencies between attributes and the target output, the Spearman's rank correlation coefficient [Spe04] is used. Only the codes that provide the highest correlations to the target output will be further considered in order to define the relations between attributes and to mine the interesting RARs.

2.2.2 Data Pre-processing

After a set containing types of codes that are relevant in defining the relational association rule model was identified (Section 2.2.1), another statistical analysis is carried out on the training data sets in order to reduce the dimensionality of the input data, by eliminating attributes which do not influence the output value. To determine the dependencies between attributes and the target output, the Spearman's rank correlation coefficient is used. The goal of this step is to remove from the attribute set those attributes (nucleotides at a certain position, in our example) that have no significant influence on the target output, i.e. are slightly correlated with it (the absolute value of the correlation is below a small positive threshold).

2.2.3 Building the Classifier

At this step, the interesting RARs are discovered in the training data sets. The classification model consisting of interesting RARs discovered in the training data sets will be further used to classify all test instances.

The training consists in applying the $DRAR$ algorithm to determine two sets of RARs having a minimum $support$ and $confidence$: a positive RAR set, denoted RAR_+, from the data set with the positive instances and a negative RAR set, denoted RAR_-, from the data set containing non-promoters.

2.2.4 Classification

After the training was completed and the $PCRAR$ classifier was built, when a new DNA sequence S has to be classified, we calculate the score $P_+(S)$ (simply denoted as P_+), a value specifying whether S contains a promoter region and $P_-(S)$ (simply denoted as P_-), another score, which indicates that S does not contain a promoter region. We propose two techniques for computing these scores. The first one depends solely on the total number of generated association rules (positive and negative) and on the number of rules that the new sequence verifies or not, without taking into consideration the confidences of the rules. The second one, on the other hand, is based on the confidences of the generated RARs. We have started from the intuition that the more relevant the RARs detected in the training data, the more precise the scores will be. That is why our main focus is toward identifying accurate and significant relations in the training data.

2.2.4.1 Score computation based on the number of rules

The score P_+ to determine if an instance should be classified as a *positive* one is:

$$P_+ = \frac{n_+ + m_-}{|RAR_+| + |RAR_-|} \tag{2.1}$$

where by n_+ we denote the number of positive RARs that are verified by the instance and m_- denotes the number of negative rules that are *not* verified by the given instance.

If $P_+ \geq 0.5$ then the instance S will be classified as a *positive* instance, otherwise it will classified as a *negative* instance.

Similarly, the score P_- to determine if an instance should be classified as a *negative* one can be computed. However, this step can be skipped as it can be easily proven that the results provided by the *PCRAR* classifier are logically consistent, meaning that for a given instance S, $P_+ + P_- = 1$.

The algorithm using the score computation described above is *PCRAR*.

2.2.4.2 Score computation based on the confidence of the rules

A second method for computing the scores is based on the confidences of verified/unverified rules. The score P_+ for a DNA instance S is computed as:

$$P_+ = \frac{1}{2}\left(\frac{s_+(S)}{s_+} + \frac{sn_-(S)}{s_-}\right) \tag{2.2}$$

where $s_+(S)$ is the sum of confidences of the rules from the set RAR_+ which are verified by S, $sn_-(S)$ is the sum of confidences of the rules from RAR_- which are *not* verified by S, while s_+ and s_- are the total sums of the confidences of all the rules from the sets RAR_+ and RAR_-, respectively.

As for the other case, if $P_+ \geq 0.5$ then the instance S will be classified as a *positive* instance, otherwise it will classified as a *negative* instance. The score P_- can be computed analogously, but this step can be skipped as it can be proven that for a given instance $P_+ + P_- = 1$.

The algorithm using the score computation described above is called *BRSC* (Binary Rules, with Score computation based on the Confidence of the rules).

Remark 1 *In the case of PCRAR it is enough to generate only the binary interesting relational rules, as a certain rule with a length greater than two is verified if its binary subrules are verified. This significantly reduces the training time of the classifier. Where confidence is also involved in the score calculation, we can either consider only binary rules or rules of any length. BRSC uses binary rules. Another algorithm, called* KRSC *(K-length Rules, with Score computation based on the Confidence of the rules), which uses rules of any length is presented in the following.*

2.2.4.3 *K*-length Rules Generation

As another extension of *PCRAR*, we propose the generation of rules of any length k, the maximum length being the number of attributes of an instance (for any instance S, let us denote its number of attributes $|S|$). A k-length rule is verified by an instance if all its $k-1$ binary subrules are verified. As soon as all k-length rules ($k \in \{2, 3, \ldots, |S|\}$) have been generated, when a new DNA sequence must be classified, we compute the scores for this sequence as described in Subsection 2.2.4.2.

2.3 Experimental Evaluation

In this subchapter we aim at experimentally evaluating our approach for promoter sequences recognition using RARs, as well as providing a comparison with other existing similar approaches.

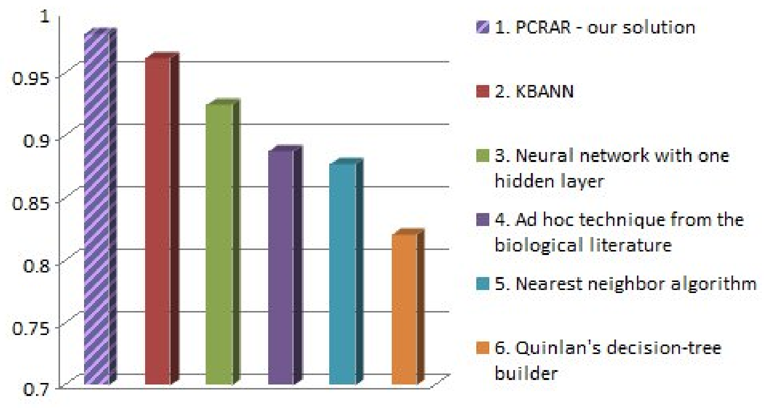


Figure 2.1: Comparative results. Our *PCRAR* classifier proves to outperform other classifiers in literature [TSN90] that have been applied to the promoter sequences prediction problem.

2.3.1 Data Set

The data set we used to test the efficiency of the classifiers is entitled "E. coli promoter gene sequences (DNA) with associated imperfect domain theory". This data set was taken from the UCI Repository [FA10] and contains a set of 106 promoter and non-promoter instances, each one having a length of 57 nucleotides. Half of the sequences represent positive instances and half are negative ones. We have considered this data set in our experimental evaluation for two reasons: first, because information about this data set (including its previous usage) are publicly available; second, as classifiers were already developed and validated on this data set, comparisons of our *PCRAR* classifier with the models existing in the literature can be conducted.

2.3.2 Results and Discussion

The methodology and algorithms described in Subchapter 2.2 is applied to the considered data set and the results are presented in the following.

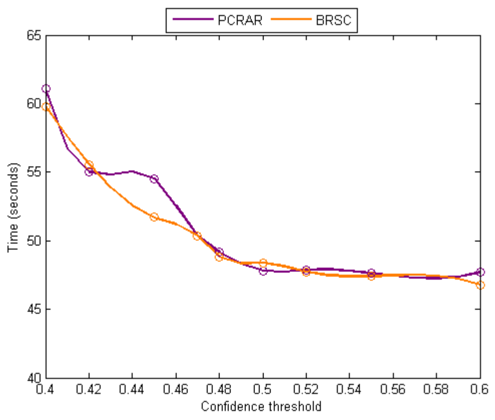
2.3.2.1 *PCRAR* - Results and analysis

We executed the *PCRAR* classification algorithm introduced in Subchapter 2.2 with minimum support threshold $s_{min} = 0.9$ and different values for the minimum confidence threshold c_{min} : $\{0.6, 0.55, 0.52, 0.5, 0.48, 0.47, 0.45, 0.42, 0.4, 0.38, 0.36\}$. For evaluating the performance of our approach, we have used the data set described in Section 2.3.1 and a cross-validation using a "leave-one-out" methodology was applied. The best result was obtained for a confidence threshold of **0.4**, for which a classification error of 0.018867 (**2/106**) was reported after the validation was completed in **64.430** seconds. Compared to the classifiers already applied in the literature for promoter sequences recognition [TSN90], the classifier introduced in this paper outperforms the *best* existing classifier for promoter sequences prediction. This comparison is illustrated in Figure 2.1 [CBC12]. In this figure, the hatched bar indicates the performance of our *PCRAR* classifier. Another advantage of this approach compared to the existing approaches is that the training step of *PCRAR* is very fast, as it is enough to discover only binary relational association rules.

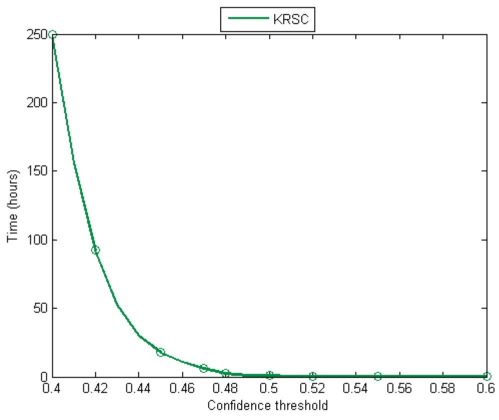
2.3.2.2 *PCRAR* and its extensions - Comparative results

The other two algorithms presented in Subsections 2.2.4.2 and 2.2.4.3 were tested on the same data set and we compared them with *PCRAR*.

Concerning the number of generated RARs (for the entire positive and negative data sets), both *PCRAR* and *BRSC* determine binary rules, that only depend on the input data set, therefore these two algorithms will always generate the same number of rules, for a certain value of the confidence threshold. However, the number of rules generated by *KRSC* will be significantly greater, as this



(a) Comparative running times: *PCRAR* and *BRSC* (in seconds). It can be observed that the running times of the two algorithms are quite similar, the maximum difference being for the minimum confidence of 0.45.



(b) Running time for *KRSC*. It can be noticed that the confidence-time function is exponential.

Figure 2.2: Comparative running times for the three RAR based algorithms.

algorithm determines rules of any length, starting from the set of generated binary rules. The maximum possible length for a k-length rule is $k = 57$ (the number of attributes in an instance), but the maximum length of the rules that were actually generated was $k = 8$.

The best result obtained by *BRSC* is the same as the one obtained by the *PCRAR*: **104** correctly classified instances, out of 106, for the minimum confidence $c_{min} = 0.4$. *KRSC*, on the other hand, has proven a worse performance, as the best obtained result is **97** correctly classified instances, out of 106 (an error of 0.084905), for a confidence of 0.6.

We will now refer to the running times of the algorithms. It is important to know that these are actually validation times, i.e. overall times in which each classifier performs the validation (this including the training time). Both the algorithms that only consider binary rules have very low computational times ($\sim$ 2 minutes), as illustrated in Figure 2.2a [Boc12c]. The one that generates rules of any length clearly runs much slower (from 5 minutes to 250 hours), as can be seen in Figure 2.2b [Boc12c].

The obtained results demonstrate that the algorithms that generate and use only binary RARs perform better than the one generating rules of any length, both in terms of classification accuracy and validation times. This leads us to the conclusion that, for the considered problem, binary rules are sufficiently relevant in order to obtain a good classification of a DNA sequence as a *promoter* or a *non-promoter*.

2.4 Conclusions and Further Work

This chapter introduced a classification model based on relational association rules discovery for promoter sequences prediction, presented in our original paper [CBC12]. The experimental evaluation of the proposed model has shown that our classifier is better than the classifiers already applied for the considered problem, indicating the potential of our proposal. We have also introduced two extensions to the RARs classification model, presented in the original papers [Boc12c], [Boc12b]. We experimentally evaluated and compared the three algorithms.

The good performance of the classification model introduced in this chapter leads us to the conclusion that machine learning models and data mining techniques are significant soft computing tools able to recognize patterns in biological data, that are hard to be identified using conventional techniques.

Further work will be made in order to identify and consider in the RARs discovery different types of relations between the nucleotides from a DNA sequence. Also, we intend to improve the accuracy of the RARs based classifiers by using supervised learning to identify the most appropriate values for the used parameters. Directions to hybridize our classification model, by combining it with other machine learning based predictive models [Mit97] will be further considered.

Chapter 3

New Reinforcement Learning Based Approaches in Bioinformatics

This chapter starts by introducing the main aspects of reinforcement learning and then continues to present our original contributions, i.e. two new reinforcement learning based models and a distributed reinforcement learning based approach, used to offer solutions to two important problems in bioinformatics: the DNA fragment assembly problem (Subchapter 1.4) and the protein tertiary structure prediction problem (Subchapter 1.5).

The two new reinforcement learning based models and the distributed reinforcement learning based approach presented in this chapter, as well as their applications within the field of bioinformatics are original works published in [CBC13, CCB13, BCC11a, CBC11a, BCC11b, CBC11c, CBC11b, CBC11d].

3.1 Reinforcement Learning. Background

Reinforcement Learning (RL) [SB98] is an approach to machine intelligence that combines two disciplines to successfully solve problems that neither discipline can address individually: *dynamic programming* and *supervised learning*. In the machine learning literature, RL is considered to be the most reliable type of learning, as it is the most similar to human learning.

RL deals with the problem of how an autonomous agent that perceives and acts in its environment can learn to choose optimal actions to achieve its goals [Mit97]. In a RL scenario, the learning system selects actions to perform in the environment and receives *rewards* (or *reinforcements*) in the form of numerical values that represent an evaluation of the selected actions [PU98]. The computer is simply given a goal to achieve and it learns how to achieve that goal by trial and error interactions with its environment. The learner is not told which actions to take, but instead must discover which actions yield the highest reward by trying them. In a RL task the agent's goal is to maximize the sum of the reinforcements received when starting from some initial state and proceeding to a terminal state.

A RL problem has three fundamental parts [SB98], as follows. *The environment* is represented by "states". By interactions with the environment, a RL system will learn a function that maps states to actions. The goal of the RL system is defined using the concept of a *reinforcement function*, which is the function of reinforcements the agent tries to maximize. This function maps states or state-action pairs to reinforcements. The agent will learn to perform those actions that will maximize the total amount of reward received on a path from the initial state to a final state [HH96]. *The value (utility) function* is a mapping from states to state values. The value of a state indicates the desirability of the state and is defined as the sum of rewards received on a path from that state to a final state. The agent will learn to choose the actions that lead to states having a maximum utility [HH96].

A general RL task is characterized by four components: a *state space* $\mathcal{S}$ that specifies all possible configurations of the system; an *action space* $\mathcal{A}$ that lists all available actions for the learning agent to perform; a *transition function* δ that specifies the possibly stochastic outcomes of taking each action in any state; a *reward function* that defines the possible reward of taking each of the actions.

The two basic concepts behind reinforcement learning are trial and error, search and delayed reward [CK91]. The agent's task is to learn a control policy, $\pi : \mathcal{S} \to \mathcal{A}$, that maximizes the expected sum E of the received rewards, with future rewards discounted exponentially by their delay, where E

is defined as $r_0 + \gamma \cdot r_1 + \gamma^2 \cdot r_2 + ...$ ($0 \leq \gamma < 1$ is the discount factor for the future rewards).

There are two basic RL designs to consider. The agent can either learn a *utility function* (U) on states (or states histories) and use this to select actions that maximize the expected utility of their outcomes, or it can learn an *action-value function* (Q) giving the expected utility of taking a given action in a given state. The latter is called *Q-learning.*

3.1.1 Q-learning - An Action-Value Function Reinforcement Learning Algorithm

One of the most used RL algorithms is *Q-learning*, in which the agent learns an action-value function. Rather than finding a mapping from states to state values, Q-learning finds a mapping from state/action pairs to values (called *Q-values*). Instead of having an associated value function, Q-learning makes use of the Q-function. In each state, there is a Q-value associated with each action. The definition of a Q-value is the sum of the (possibly discounted) reinforcements received when performing the associated action and then following the given policy thereafter. An *optimal Q-value* is the sum of the reinforcements received when performing the associated action and then following the optimal policy thereafter.

If $Q(s,a)$ denotes the value of doing the action a in state s, $r(s,a)$ denotes the reward received in state s after performing action a and s' represents the state of the environment reached by the agent after performing action a in state s, the equation that is most often used for the value iteration update is the following [WD92]:

$$Q(s,a) = (1-\alpha) \cdot Q(s,a) + \alpha \cdot (r(s,a) + \gamma \cdot \max_{a'} Q(s',a')). \tag{3.1}$$

where γ is the discount factor for the future rewards and $\alpha \in [0,1]$ is the learning rate.

The Q-learning algorithm has been proven to converge to an optimal policy and action-value function as long as all stat-action pairs are visited an infinite number of times and the policy converges in the limit to the *greedy* policy [PS94].

3.1.2 Action Selection Policies

An important aspect in reinforcement learning is maintaining an equilibrium between *exploitation* and *exploration* [Thr92]. The agent has to be able to accumulate a lot of reward, by choosing the best experienced actions, but it must also explore its environment, by trying new actions (maybe not the optimal ones) that may lead to better future action selections. Several rules (policies) for choosing actions in order to make transitions among states during the learning process exist in the literature.

The *greedy* policy implies that the learning agent choose the highest-valued action in each state [SB98]. Another method, which balances the exploration of new states with exploitation of current knowledge, is ϵ-*Greedy* [SB98]. An agent following this policy will choose, in most cases (with probability $1-\epsilon$), the action having the highest reward, but with a small probability (ϵ), it will explore new actions from a given state by selecting an action at random, uniformly, independently of the action-value estimates. This policy also has a disadvantage, specifically the fact that when selecting a random action the worst action is as likely to be chosen as the second best one. A way to counter this disadvantage is to use a policy that chooses better actions more often - the *softmax policy.* This selection mechanism has the advantage that actions are ranked according to their value estimates and each action is chosen with a probability computed using its value [SB98].

3.1.3 Eligibility Traces

Eligibility traces were firstly introduced in [Klo72] and they are a basic mechanism used in RL for handling delay [SS96]. The idea is that each time a state is visited it is marked by a trace, which then gradually decays over time, exponentially, according to a decay parameter λ ($0 \leq \lambda \leq 1$) and to the discount rate parameter γ. The trace makes the state *eligible* for learning [SS96].

There are two types of possible implementations for eligibility traces. The first one is *accumulating eligibility traces* - the trace increases each time a state is visited. States that are visited more recently and more often are assigned more credit. The second type is *replacing eligibility traces* - each time a state is visited its trace is reset to 1, disregarding the previous trace information.

3.2 Proposed Reinforcement Learning Based Models. Theoretical Considerations

In this subchapter we aim to introduce two new RL based models and a distributed RL based approach, which will subsequently be used to tackle three important problems in bioinformatics.

From a computational perspective, each of the three problems we are aiming to approach using RL techniques (the DNA fragment assembly problem - Subchapter 1.4, the protein tertiary structure prediction problem - Subchapter 1.5 and the temporal ordering problem - Subchapter 1.6) is a combinatorial optimization, NP-complete problem. Another characteristic shared by all three problems is that each one can be regarded as a *generalized permutation finding problem*, meaning that the solution can be encoded into a generalized permutation of a given set of objects, which satisfies certain conditions and optimizes a given objective function. Here, by *generalized permutation*, we denote all *ordinary permutations* (in which each object is distinct) joint with all *permutations with repetitions*, in which multiple copies of the same object are allowed.

Let us give a general definition of an optimum permutation finding problem. We are given an input set of n objects and a generic objective function, defined on the set of all possible generalized permutations of the input objects, which associates to each permutation a real value, indicating its quality and relevance. The goal is to determine a generalized permutation σ of $\{1, 2, \dots, n\}$ that optimizes the objective function of the sequence of objects considered in the order given by σ. We denote by m the length of the permutation σ, where $m \geq n$ ($m = n$ for the cases in which the objects cannot be repeated).

3.2.1 Path Finding Model

In the following, we introduce a first RL based model, called the *path finding model.* Its main idea is based on constructing a generalized permutation of the given objects that optimizes the objective function by starting from an empty sequence and iteratively adding the most appropriate objects until the permutation is completely formed.

The environment in the path finding model may be visualized as a tree, containing $\frac{n^{m+1}-1}{n-1}$ vertices, each vertex corresponding to a state. The *initial state* is represented by the root. The action space consists of n actions corresponding to the n possible values $1, 2, \dots, n$ used to represent a solution. At a given moment the agent can cause a transition from a state to any of the n successor states, by executing one of the n possible actions. The transitions between the states are equiprobable, the transition probability being equal to $1/n$. The reward the agent receives for each transition is problem related and in most cases it is defined so as to optimize the objective function.

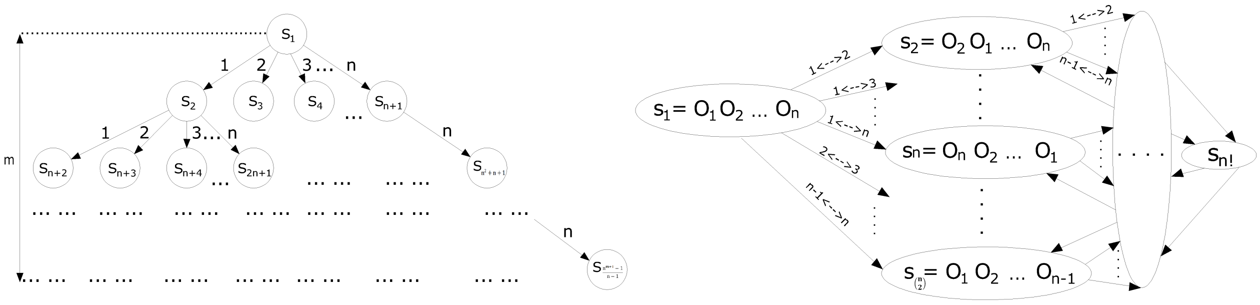
A graphical representation of the states space associated to path finding model is given in Figure 3.1a. From the initial state s_1, the agent will choose one of the n actions to transition to the next state. This process will be repeated for each new state, causing the agent to "descend" one level at a time (on the tree), until it reaches a *final state* - one of the states positioned on the last level. In this tree-like environment, a path to the final state consists of distinct vertices (states) in which each adjacent pair of vertices is linked by an arc (action). The sequence of actions obtained following the transitions between the successive states from this path will be referred to as the *action configuration* associated to the path and it gives a sequence (or a generalized permutation) of the input objects.

For each generalized permutation problem, during some training episodes, the agent will be trained to find a path having the optimum associated value of the objective function, using the Q-learning algorithm. After the RL training process, the agent learns to execute those transitions that maximize the sum of rewards received on a path from the initial to a final state.

3.2.2 Permutation Model

The second RL based model, called the *permutation model*, is presented below. This model particularly refers to ordinary permutations, in which all elements must be distinct (i.e. $m = n$). The idea is to reach a permutation of objects having an optimum value of the objective function by starting from the identical permutation and iteratively refining it (by deriving new permutations) until reaching a final permutation having the associated values of the objective function sufficiently close to a goal value.

In this case, each state represents a possible solution, i.e. a permutation of the input objects. The environment is graphically represented in Figure 3.1b. It consists of $n!$ states. The *initial state* is the



(a) Environment for the path finding model. The circles represent states and the transitions between states are indicated by arrows labeled with the action leading from one state to another.

(b) Environment for the permutation model. O_i denotes the i-th object in the input data set. The states are represented within the ellipses and the transitions are indicated by arrows labeled with the action leading from one state to another.

Figure 3.1: Graphical representation of the environments for the two RL based models.

identical permutation of the objects and we consider s state *final (terminal)* if the associated value of the objective function is sufficiently close to a goal value. A priori knowledge about the problem is needed to be able to define a final state. To transition from one state to another, the agent may choose one the $\binom{n}{2} = \frac{n(n-1)}{2}$ actions. Each action is a pair of indices specifying that the objects located at those indices in the current state will be interchanged to reach the successor state. The transitions between the states are equiprobable. The reward function will be based on the objective function for a sequence of objects and must be defined for each specific problem so as to optimize this function.

Unlike for the path finding model, in this case we are not interested in the sequence of actions the agent determines in order to reach a final state, but rather in the final state of the environment after the agent executes the learned policy. This final state, representing a permutation of the initial objects, should have an associated value very close to the optimum value for the defined objective function.

3.2.3 Distributed Reinforcement Learning Based Model

In order to speed up the trainig process, we extend the proposed RL based models towards a distributed RL approach. We propose a kind of concurrent Q-learning approach, where multiple cooperative agents learn to coordinate in order to find the optimal policy in their environment.

In our distributed architecture we have two types of agents: *local agents*, which run in separate processes or threads and are trained using the Q-learning algorithm; a *supervisor agent*, which supervises the learning process and synchronizes the computations of the individual local agents. It keeps a blackboard [GBF$^+$07] that stores the global Q-values estimations. This supervisor updates the global Q-value estimation only if the new estimation received from the local agents are better than the Q-values existing in the blackboard.

During some training episodes, the individual local agents will experiment some paths from the initial to a final state, updating the Q-values estimations according to the Q-learning algorithm (Section 3.1.1) and using either of the two models presented earlier in this section (the path finding model or the permutation model). After the training of the multi-agent system has been completed, the solution learned by the supervisor agent is constructed by starting from the initial state and following the *greedy* mechanism until reaching a final state.

3.2.4 A New Intelligent Action Selection Policy

We introduce a new intelligent action selection policy aiming to better guide the reinforcement learning agent towards good solutions. This action selection mechanism is derived from the ϵ-Greedy mechanism (Section 3.1.2) and it uses a one step look-ahead procedure in order to better guide the exploration of the search space. When selecting an action from a given state, the following selection mechanism is used: with probability $1 - \epsilon$ select the action that maximizes the Q-value of the reached neighboring state (*ϵ-Greedy selection*); with probability ϵ, select the action that optimizes the objective function corresponding to the current configuration (*look-ahead*).

3.3 Reinforcement Learning Based Model for the DNA Fragment Assembly Problem

In this subchapter, the three RL based models that were introduced in Sections 3.2.1, 3.2.2 and 3.2.3 are adapted to approach the DNA fragment assembly (FA) problem. This problem was presented in Subchapter 1.4 and it refers to reconstructing an original DNA sequence from smaller DNA fragments.

3.3.1 Methodology

Let us consider, in the following, that $\mathcal{S}eq$ is a DNA sequence and $\mathcal{FS} = \{F_1, F_2, \ldots, F_n\}$ is a set of fragments. Each fragment is a small DNA subsequence and fragments contain overlapping regions. The FA problem consists of determining the order in which these fragments have to be assembled back into the original DNA molecule, based on common subsequences of fragments. Consequently, the FA problem can be viewed as the problem of generating a permutation σ of $\{1, 2, \ldots, n\}$ that optimizes the performance of the alignment $\sigma_{\mathcal{FS}} = (F_{\sigma_1}, F_{\sigma_2}, \ldots, F_{\sigma_n})$ $(n > 1)$. The objective function that must be maximized is a performance measure PM, which sums the overlap scores over all adjacent fragments [PFB95].

First, we adapted and applied the path finding model to solve this problem. The state and action spaces, as well as the transition function remain the same. We offer two possible definitions for the reward function, both depending on the performance measure for an alignment. The first reward function, as defined in Formula (3.2), illustrates situations when feedback is given at the end of each trial episode. This function can be modified so as to give feedback to the agent after each transition to a new state, even if the state is not final. Therefore, the second definition of the reward function is given in Formula (3.3).

$$r(\pi_k) = \begin{cases} PM(F_{a_\pi}) & if \;\; k = n \\ \tau & otherwise \end{cases} \quad (3.2) \qquad r(\pi_k) = \begin{cases} 0 & if \;\; k = 0 \; or \; k = 1 \\ w(F_{a_{\pi_{k-1}}}, F_{a_{\pi_{k-2}}}) & otherwise \end{cases} \quad (3.3)$$

where by $r(\pi_k)$ we denote the reward received by the agent in state π_k, $k \in \{1, \ldots, n\}$, after its history in the environment is $\pi_0 = s_1, \pi_1, \pi_2, ...\pi_{k-1}$, $a_\pi = (a_{\pi_0} a_{\pi_1} \ldots a_{\pi_{n-1}})$ is a possible action configuration and $w(a, b)$ is the similarity score of the fragments a and b.

3.3.2 Experimental Evaluation and Results

The computational experiments are performed on a toy example, a small DNA sequence belonging to the bacterium *Escherihia coli (E. coli)*, as well as on a human DNA sequence.

3.3.2.1 Experiment 1 - Toy example

We first tested our approach on the example described in Subchapter 1.4. The DNA sequence is $TTACCGTGC$ and the set of fragments is: $F_1 = ACCGT$, $F_2 = CGTGC$, $F_3 = TTAC$, $F_4 = TACCGT$. The correct order of fragments which gives the original DNA sequence is: $F_3F_4F_1F_2$.

We have trained the agent using as reward function the function defined in Formula (3.2). We used the ϵ-Greedy action selection mechanism and 240 training episodes. Two optimal solutions were reported after the training of the agent was completed: $F_3F_4F_1F_2$ and $F_2F_1F_4F_3$, both of them having the maximum associated performance.

On this toy example, we have also applied the distributed RL model introduced in Section 3.2.3. We used two local agents, each one using a path finding Q-learning based approach. The two optimal solutions were reported by the supervisor agent after the training of the local agents.

3.3.2.2 Experiment 2 - E. coli small DNA sequence

In this second example, we have chosen a small section of DNA belonging to the bacterium *Escherichia coli (E. coli)*. The DNA sequence contains 25 nucleotides: $TACTAGCAATACGCTTGCGTT$ $CGGT$. Using the Perl scrips from [ZCY$^+$11] we obtained 10 fragments, each having 8 nucleotides. These are ordered in the following way to form the original DNA sequence: $F_6F_3F_{10}F_5F_7F_9F_1F_8F_2F_4$.

Experiments were made with both the reward functions defined in Section 3.3.1 and using both the ϵ-Greedy policy, as well as the intelligent action selection policy introduced in Section 3.2.4. The solution reported after the training of the agent was completed is the correct one, having the maximum performance measure, for all the tests.

In the first series of tests we used the reward function defined in Formula (3.2) and the ϵ-Greedy selection policy, with the ϵ parameter set to 0.8. The solution, i.e. the alignment having the optimal performance measure, is reached after $486 \cdot 10^3$ episodes, on average. Second, we tested the RL based model using the reward function defined in Formula (3.3) and the ϵ-Greedy policy, with the same value of ϵ. As the agent is rewarded after each step, the learning happens faster. As a result, the correct fragment alignment is obtained after an average of $334 \cdot 10^3$ episodes. Lastly, the RL based model was tested using the reward function defined in Formula (3.3) and the intelligent action selection policy introduced in Section 3.2.4. By employing this selection mechanism, the agent reaches the optimal solution even faster, on average, after less than 10^3 episodes.

3.3.2.3 Experiment 3 - Human DNA

To evaluate our algorithm on larger sequences, we have chosen a human DNA sequence. It was taken from NCBI (National Center for Biotechnology Information) [LAR+10]. The sequence refers to a "Human MHC class III region DNA with fibronectin type-III repeats" and it has a length of 3835 nucleotides. We used a software called GenFrag [EB96] to generate 20 fragments for the above mentioned sequence. The average length of a fragment is 766 nucleotides.

In order to avoid the grouping together of fragments which have long overlaps, but should not be neighbors in the final assembly, in this situation we use a different performance function, which was introduced by Parsons *et. al* [PFB95] and that penalizes solutions in which strong overlaps occur between fragments that are further apart in the alignment. This function must be minimized.

Considering the performance measure mentioned above, the reward function is also modified, being defined as in Formula 3.4:

$$r(\pi_k) = \begin{cases} 0 & \textit{if} \quad k = 0 \textit{ or } k = 1 \\ -2 \cdot \displaystyle\sum_{i=0}^{k-2} (k-1-i) \cdot w(F_{a_{\pi_i}}, F_{a_{\pi_{k-1}}}) & \textit{otherwise} \end{cases} \tag{3.4}$$

where by $r(\pi_k)$ we denote the reward received by the agent in state π_k, $k \in \{1, \ldots, n\}$, $a_\pi = (a_{\pi_0} a_{\pi_1} \ldots a_{\pi_{n-1}})$ is a possible action configuration and $w(a, b)$ is the similarity score of the fragments a and b.

It can be proven that the total sum of rewards is actually the opposite of the performance measure. The reward will be maximized, therefore the performance function of an alignment will be minimized.

We evaluated the obtained alignment in terms of number of contiguous sequences. The recovered optimal permutation composes a single contiguous sequence which will then be used to obtain the target DNA. Our RL based algorithm was run 5 times and it converges to this solution quite rapidly, after $32 \cdot 10^3$ episodes, on average.

3.3.3 A Comparison of the *Path Finding* and the *Permutation* Models

To compare the performances of the two models, we consider the same section of DNA from the bacterium *Escherichia coli (E. coli)*, but in this case we generated only 8 fragments, which compose the DNA sequence as follows: $F_4F_7F_5F_6F_8F_3F_2F_1$.

Both RL based models are applied to find the optimum permutation of the fragments. The path finding model is applied similarly to the previous two experiments, using as reinforcement function the one defined in Formula (3.3). Regarding the permutation model, the state and action spaces as well as the transition are the ones presented in Section 3.2.2. The reinforcement function associated to a transition from a state $s_j = \sigma^j_{\mathcal{FS}}$ to a state $s_l = \sigma^l_{\mathcal{FS}}$ ($j,\ l \in \{1, \cdots, n!\},\ l \neq j$), by executing action $a_k, 1 \leq k \leq N_a$, will be:

$$r(s_l = F_{\sigma^l} | s_j, a_k) = \begin{cases} PM(\sigma^l_{\mathcal{FS}}) - PM(\sigma^j_{\mathcal{FS}}) & \textit{if } s_l \textit{ is non-terminal} \\ PM(\sigma^l_{\mathcal{FS}}) - PM(\sigma^j_{\mathcal{FS}}) + PM(s_0) & \textit{otherwise} \end{cases} \tag{3.5}$$

where s_0 is the identical permutation.

We compare the two models by running the corresponding implementations, using the Q-learning algorithm [SB98] in conjunction with three action selection mechanisms: ϵ-Greedy, softmax (Section

3.1.2) and the intelligent action selection mechanism (Section 3.2.4). The permutation model proved to outperform the path finding approach, for the considered case study, when inspecting the number of epochs, as well as the the computational time. Still, in the case of the permutation model, we note that because it is difficult to determine a final state, we need a priori information about the possible values of the performance measure for permutations.

For the permutation model the number of states of the environment is smaller than for the path finding approach. However, an important drawback of this model is the fact that a priori knowledge about the problem is needed in order to define a final state and this information is not available in all types of situations. Therefore, the path finding approach is more general and, as the results it obtains are good both in terms of accuracy and in terms of computational time, we conclude that it is better.

3.3.4 Discussion

We have experimentally evaluated each RL based model that we introduced on several DNA sequences, using the Q-learning algorithm with two different reinforcement functions and three action selection policies with different values for the parameters.

Regarding the path finding RL based approach applied for solving the DNA fragment assembly problem, the training process during an episode has a time complexity of $\theta(n)$ (n is the number of fragments), when the Q-learning algorithm is used with the ϵ-Greedy action selection policy. For the case when the agent performs an intelligent action selection mechanism (the *look-ahead* procedure), the training process during an episode has a time complexity of $\theta(n^2)$. Consequently, assuming that the number of training episodes is k, the overall complexity of the algorithm for training the agent is $\theta(k \cdot n)$ or $\theta(k \cdot n^2)$, depending on the used policy.

We have introduced two reinforcement functions for an agent who learns using the path finding model. The function rewarding the agent after each transition leads to faster learning than the function which offers the reward only at the end of a trial episode. As soon as the intelligent action selection policy (Section 3.2.4) is introduced, the learning is even more accelerated.

To avoid considering a very large number of episodes for obtaining a solution, a distributed RL based model was introduced. The main advantage of the distributed approach is that, by using multiple agents during the training step, the overall computational time is reduced. Still, the problem that has to be further investigated is how to preserve the accuracy of the results in the distributed approach.

3.4 Reinforcement Learning Based Model for the Protein Tertiary Structure Prediction Problem

This subchapter describes how two of the RL based models introduced in Subchapter 3.2, more specifically the path finding (Section 3.2.1) and the distributed models (Section 3.2.3) are applied to solve the protein tertiary structure prediction (PTSP) problem, which, from a computational perspective, refers to predicting the three dimensional structure of a protein from its amino acid sequence. This problem was presented in more detail in Subchapter 1.5.

3.4.1 Methodology

To define the RL task, we use the Hydrophobic-Polar (HP) model [Dil85] (Subchapter 1.5). For a given protein, composed of amino acids (each amino acid being either hydrophobic or polar), the problem is to find the position of each amino acid in a two dimensional lattice so as to minimize an energy function.

A solution for the bidimensional HP PTSP problem, corresponding to an q-length protein $\mathcal{P}$ could be represented by a $q-1$ length sequence, where each position encodes the direction of the current amino acid relative to the previous one (L - left, R -right, U - up, D - down).

We present in the following how the path finding RL based model introduced in Section 3.2.1 is adapted and applied to solve the PTSP problem. Given that a solution can be seen as a sequence composed of the four symbols (L, R, U, D) expressing directions of amino acids, the PTSP problem can be regarded as the problem of generating a generalized permutation π that minimizes the energy function in the HP model. In this case, the permutation is not an ordinary one, but is a permutation with repetitions, as each of the four symbols may appear as many times as necessary. Hence, we remark that the objects mentioned in Subchapter 3.2 are here represented by the four directions, i.e. $n = 4$ and the length of a permutation m is one unity less the number of amino acids forming the protein's primary sequence.

The state and action spaces, as well as the transition function are similar to the ones presented in Section 3.2.1. The reward function is defined so as to ensure the minimization of the protein's energy. Therefore, after a transition to a non-terminal state the agent receives a small positive constant and when transiting to a final state the agent receives as reward the opposite value of the energy function associated to the obtained configuration. In this way, by trying to maximize the sum of rewards, the agent actually tries to minimize the energy function.

3.4.2 Experimental Evaluation and Results

The path finding and the distributed RL based models (Section 3.2) are experimentally evaluated using some artificially generated HP protein sequences, as well as a benchmark generally used for the bidimensional protein structure prediction.

3.4.2.1 Experiment 1

We consider an easy to follow example, a HP protein sequence $\mathcal{P} = HHPH$, consisting of four amino acids, i.e $q = 4$. The agent was trained using the Q-learning algorithm. We used the ϵ-Greedy action selection mechanism was used, with $\epsilon = 1$ and 100 training episodes.

The solution reported after the training of the agent was completed has a minimum associated energy of -1. This optimal solution is the *configuration* (ULD). This solution is actually equivalent to the following configurations: (RUL), (DRU) and (LDR), which are all its rotations around the fixed point in the lattice where the first amino acid is mapped. Also, this solution is equivalent to the configuration (URD), which is its mirrored version, as well as with the mirrored equivalent rotations (LUR), (RDL) and (DLU). The agent may reach any of these solutions, which are all correct and have the minimum associated energy of $E^* = -1$.

On this simple example, we have also applied the distributed RL model introduced in Section 3.2.3. Two local agents are trained during 40 episodes, using a path finding approach. After their training is completed, the solution reported by the supervisor agent is the same configuration: (ULD).

3.4.2.2 Experiment 2

In this second experiment the protein is composed of $q = 11$ amino acids: $\mathcal{P} = HHHPHPPPPPH$. This example is taken from [Chi10] and the minimum value of the energy function is $E^* = -2$.

The agent was trained during 10^5 training episodes and using the ϵ-Greedy action selection mechanism, with $\epsilon = 0.8$, to allow the exploration of the search space, but also the exploitation of the learned Q-values. The agent needs, on average, $43 \cdot 10^3$ epochs to obtain a configuration having the minimum energy. The reported results are averaged over 5 runs of the algorithm. There are several optimal solutions (having an energy value of -2) reported by the agent after the training was completed: $(DRDLDLULUR)$, $(DDRUUULLDD)$, $(DLDRDRRUUL)$, $(RURDRRDLLL)$, $(ULURRRDDLU)$, $(URULLLDDRU)$, $(ULURURRDLD)$, together with their rotations and their mirrored versions.

3.4.2.3 Experiment 3

In this third example, we consider a bidimensional HP protein instance $\mathcal{P} = HPHPPHHP HPPHPHHPPHPH$, consisting of twenty amino acids, i.e. $q = 20$. This is a benchmark instance generally used for the two-dimensional HP PTSP problem. It can be found in [UM93] and its known optimal energy value is $E^* = -9$.

We have trained the agent using the same parameter setting as for the previous experiment. The only difference is that when using the ϵ-Greedy action selection mechanism we started with $\epsilon = 1$ in order to favor exploration, then after the training progresses ϵ is decreased until it reaches a small value, which means that at the end of the training exploitation is favored.

Using the above defined parameters and under the assumptions that the state action pairs are equally visited during training, the solution reported after the training of the agent was completed is the configuration $(RUULDLULLDRDRDLDRRU)$. Figure 3.2 [CBC11b] illustrates this configuration.

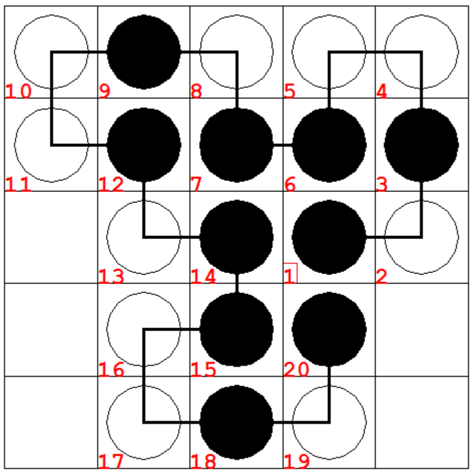


Figure 3.2: A protein configuration for the sequence $\mathcal{P} = HPHPPHHPHPPHPHHPPHPH$, of length 20. The configuration learned is $(RUULDLULLDRDRDLDRRU)$. Black circles represent hydrophobic amino acids, while white circles represent hydrophilic ones. The value of the energy function for this configuration is -9.

3.4.3 Discussion

The bidimensional protein tertiary structure prediction was approached in this subchapter, using two reinforcement learning based models that were previously introduced. These models are adapted to tackle the PTSP problem, which can be regarded, from a computational point of view, as the problem of generating an optimum permutation with repetitions, consisting of directions of the amino acids composing the protein.

Regarding the path finding RL based approach for solving the bidimensional PTSP problem, we remark the following. For the Q-learning approach that we used in conjunction with the ϵ-Greedy action selection policy, the training process during an episode has a time complexity of $\theta(n)$, where by n we denote the length of the HP protein sequence. Consequently, assuming that the number of training episodes is k, the overall complexity of the algorithm for training the agent is $\theta(k \cdot n)$. If the dimension n of the HP protein sequence is large and consequently the state space becomes very large, a function approximation method (e.g. neural network, support vector machine) should be used in order to store the Q-values estimates.

The main drawback of our approach is that a very large number of training episodes has to be considered in order to obtain accurate results and this leads to a slow convergence. In order to speed up the convergence process, further improvements will be considered.

3.5 Conclusions and Further Work

In this chapter we have introduced new reinforcement learning based models for solving a general class of combinatorial optimization problems and we have adapted and applied these models to solve two major problems in bioinformatics: the DNA fragment assembly problem (Subchapter 1.4) and the protein tertiary structure prediction problem (Subchapter 1.5). The approaches presented in this chapter are original works published in [CBC13, CCB13, BCC11a, CBC11a, BCC11b, CBC11c, CBC11b, CBC11d].

The techniques we introduced were experimentally evaluated on several data sets from the domain of each of the two problems and the obtained results were presented and analyzed. These results prove a good performance of the RL based models, which demonstrates the potential of our proposals.

We plan to extend the evaluation of all the Q-learning based algorithms for larger DNA and protein sequences, to further develop the analysis. We will also investigate possible improvements of these models by adding various local search mechanisms, by combining the softmax policy with the intelligent action selection procedure (Section 3.2.4), by modifying the action selection parameters during the training process or by considering an extension of the RL based permutation model to a distributed approach in order to solve the fragment assembly problem.

Chapter 4

New Approaches to the Biological Temporal Ordering Problem

This chapter presents two different approaches that we proposed for the biological temporal ordering (TO) problem (Subchapter 1.6). The first approach, developed in collaboration with a research team during my research internship at the University of Milano-Bicocca, tackles the TO problem from a computational point of view, focusing on biological aspects of the problem in question as well [BCG+12, BCG+13]. The second approach tackles the TO problem from a computational perspective and it focuses on applying one of the reinforcement learning (RL) models we introduced in the previous chapter to the temporal ordering problem [CBC13, Boc12a]. Furthermore, this chapter also introduces a new programming interface for solving optimization problems using RL techniques, which is applied to find solutions to the TO problem [CCB11a, CCB11b].

4.1 Temporal Ordering of Colorectal Cancer Samples through Copy Number Alteration Data

This subchapter presents a new approach to the TO problem. The method introduced here was developed during my research internship at the University of Milano-Bicocca, in collaboration with the BIMIB research group.

4.1.1 Biological Concepts

This section presents some basic biological concepts needed for a better understanding of the methodology that we propose for the temporal ordering of a copy number alterations data set.

Colorectal cancer (CRC) is the third most common type of cancer worldwide and the second most frequent cause of cancer-related death [JSXW10]. Most CRCs develop through a series of distinct morphological stages that are strongly correlated with the malfunctioning of the complex signaling networks ruling the intestinal crypt dynamics and homeostasis, which is induced by the accumulation of alterations in the function of key regulatory genes and genetic instability [VFH+88, FV90, Fra07, Net12].

Chromosomal *copy number alterations (CNAs)* refer to regions of the DNA that have either been deleted (losses or deletions) or duplicated a certain number of times (gains or amplifications) on chromosomes. These alterations may affect the function of certain genes by modifying their expression and have been associated with susceptibility or resistance to certain diseases. In cancer, chromosomal CNAs can also lead to activation of oncogenes and inactivation of tumor suppressor genes. During the progression from high-grade adenomas to invasive carcinomas, specific chromosomal aberrations become common, such as gains and deletions on certain chromosomes [ASD+10]. CNAs are directly correlated with cancer and the analysis of CNAs that happen in tumor cells could provide further insights into the process of CRC progression.

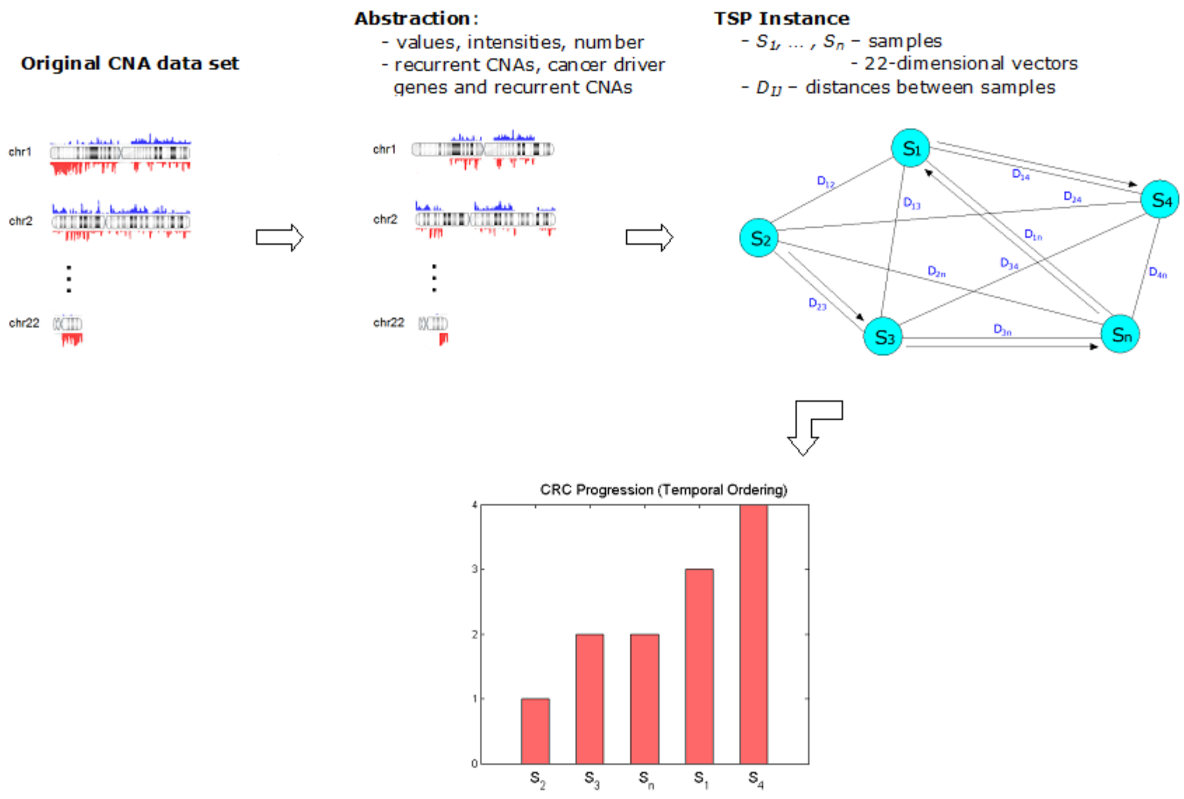

Original CNA data set
Abstraction:
- values, intensities, number
- recurrent CNAs, cancer driver genes and recurrent CNAs
TSP Instance
- S1, ..., Sn – samples
- 22-dimensional vectors
- DIJ – distances between samples
chr1
chr2
chr22
CRC Progression (Temporal Ordering)

Figure 4.1: Representation of the proposed methodology. Starting from the input data set, different subsets are defined, using abstraction mechanisms. A TSP instance is built for each new data set and finally, the solution to the TSP represents the temporal ordering of the given samples.

4.1.2 Methodology

This section introduces the methodology we propose for obtaining a temporal ordering of a set of static copy number alteration data taken from patients at different stages of colorectal cancer. Our work is more focused on a specific approach to the TO problem, previously proposed by Gupta and Bar-Joseph [GBJ08], which has been shown to work for gene expression data. The technique presented in [GBJ08] is based on the reduction of the sorting problem to the travelling salesman problem (TSP), under two biologically realistic assumptions over the gene expression data set. As we can assume that the CNAs data also fulfils these two assumptions, we develop a methodology which enables us to apply the technique on a CNAs data set.

Figure 4.1 [BCG+12] briefly illustrates our methodology, highlighting the most important steps that were used to determine a temporal ordering for a set of biological samples. These steps will be detailed in the following.

In order to capture distinct aspects of the complex CNAs phenomenon, we define several chromosome-related measures and certain filters targeting significant portions of chromosomes. We also aim to identify which of these measures performs best regarding tumour progression or whether chromosomal gains or losses, considered separately, could influence the outcome. As chromosome measures, we introduce the following notions: *value, intensity, number* and the averaged analogous: *average of the values and average of the intensities*, all these referring to alterations, deletions and amplifications. Furthermore, we propose two filtering methods to be applied on the initial data set, which could lead towards obtaining more accurate orderings: (i) recurrent CNAs - we consider those CNAs that belong to regions of the chromosomes that have suffered alterations in a higher number of different samples; (ii) recurrent CNAs, as well as CNAs belonging to regions that include at least one of the genes known to be involved in tumor progression (cancer driver genes).

To build the TSP instance, we consider the cities to be represented by the 22-dimensional samples (each dimension corresponding to one chromosome, not considering the gender-linked chromosome) and a distance matrix is used to define distances between any two samples. Two types of metrics are used: the L1 distance and the Euclidean distance.

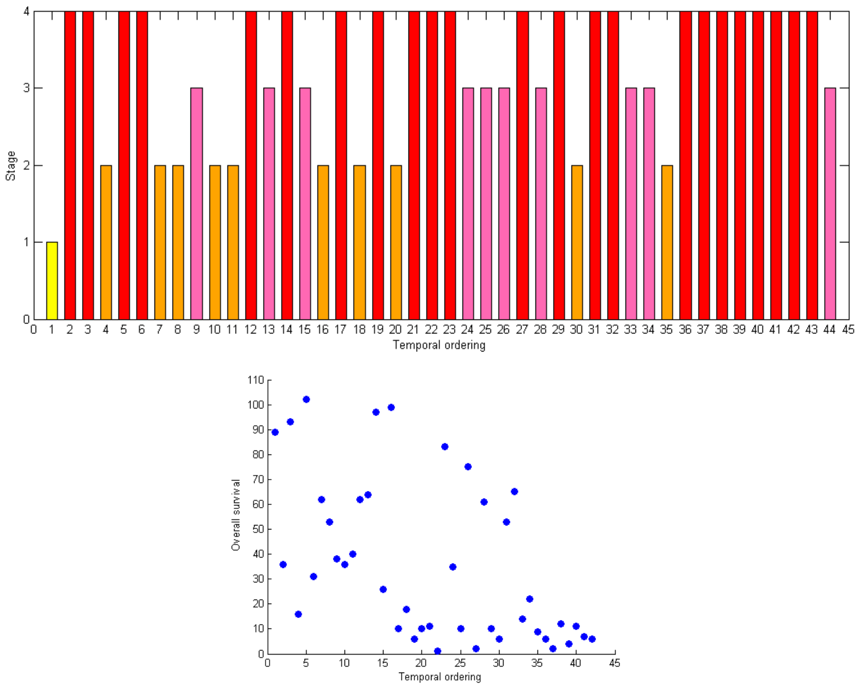


Figure 4.2: The best recovered ordering for the CRC data set. In the first figure, the ordering is plotted against the histological stages: I (yellow), II (orange), III (pink) and IV (red). In the second one, the same ordering is plotted against the overall survival time. The samples in the left part have higher survival times, as expected.

4.1.3 Experiments

This section presents the experiments developed in order to test the efficiency of our methodology. Evaluations are performed on the colorectal cancer CNAs data set.

The data set we used was taken from the study of Reid *et al.* [RGS+09]. It contains 44 biological samples extracted from patients in different stages of CRC.

Three types of tests were made, one for the initial input data set and two for the subsets obtained by applying the above mentioned filters, therefore obtaining several different orderings. As soon as the values for the chromosomes are computed for all the samples, the TSP instance is solved by using the Concorde TSP Solver [Coo11]. As a validation criterion, we used the survival time of each patient, after being diagnosed. We defined the *ideal ordering* as the one in which the first sample has the maximum, while the last one has the minimum overall survival time. Using the *squared deviation distance (SDD)* [SS05], the distance from each obtained solution to the ideal one was computed. Therefore, the orderings having smaller SDDs (with regard to the ideal ordering) were considered to be more accurate.

The best result was obtained for the test that takes into account the *CRC driver genes and recurrent CNAs*, with the chromosome measure *average of values of alterations* and for the *Euclidean distance*. Figure 4.2 [BCG+13] (bottom) illustrates this result and indicates the correlation between the obtained ordering and the survival time. The samples in the left half of the graph belong to patients whose survival times (relative to the moment of diagnosis) are higher, while the ones in the right half have lower survival times. Although in our data set the CRC histological stages are not always directly correlated to the survival time (the expectancy is that as the stage increases, the survival time should decrease), in Figure 4.2 [BCG+13] (top) we may observe that for the best ordering, there are considerably more samples in stages I and II in the left half (9 samples) than in the right one (2 samples) and there are more samples in stages III and IV in the right half (20 samples) than in the left one (13 samples). We thus observe that the order is, to a certain degree, also compatible with the histological stage. We may also remark that the sample having stage I is positioned in the first place.

4.2 Reinforcement Learning Based Model for the Biological Temporal Ordering Problem

This subchapter presents how the RL based path finding model (Section 3.2.1) can be adapted and modified to approach the biological temporal ordering problem. This problem was described in Subchapter 1.6. It refers to constructing a sorted collection of multi-dimensional biological data, collection that reflects an accurate temporal evolution of a certain biological process. Here, we restrict to considering data sets consisting of samples derived from microarray gene expression experiments.

4.2.1 Methodology

Let us consider that $\mathcal{DS}$ is the input data set, consisting of n ($n > 1$) multi-dimensional samples: $\mathcal{DS} = \{S_1, S_2, \ldots, S_n\}$, each sample being identified by a set of features. For the considered type of data, each feature is represented by one gene and has as a value a real number, measuring the expression level of the gene in question. A first step of our approach is data pre-processing. As the dimensionality of the input data can be extremely high (thousands of gene expression levels for each sample), the goal of this step is the elimination of the genes that offer no significant information in the temporal ordering process. To this end, a statistical analysis is carried out on the data set and by using the Pearson correlation coefficient [Tuf11], we select only those genes that are highly correlated with the selected supplementary biological information (which is, in the case of cancer data sets, the survival time).

From a computational point of view, the TO problem can be viewed as the problem of generating a permutation σ of $\{1, 2, \ldots, n\}$ that maximizes the overall similarity Sim of the sequence of samples considered in the order σ: $S_\sigma = (S_{\sigma_1}, S_{\sigma_2}, \ldots, S_{\sigma_n})$ ($n > 1$). We mention that in the case of the TO problem, the searched permutation is ordinary, meaning that all its composing elements must be distinct, therefore the length of the permutation is equal to the cardinality of the set of samples. The overall similarity Sim we consider in this paper sums the similarities over all adjacent samples and it has to be maximized. The overall similarity for the sequence of samples $S_\sigma = (S_{\sigma_1}, S_{\sigma_2}, \ldots, S_{\sigma_n})$ is defined as $\mathrm{Sim}(S_\sigma) = \sum_{i=1}^{n-1} \mathrm{sim}(S_{\sigma_i}, S_{\sigma_{i+1}})$, where $\mathrm{sim}(x_i, x_j)$ denotes the similarity between the multidimensional vectors x_i and x_j and is defined as $\mathrm{sim}(x_i, x_j) = Max - d_E(x_i, x_j)$. Here by d_E we denote the euclidian distance and Max is a large constant.

We aim to apply the path finding model introduced in Section 3.2.1 to obtain a path from the initial to a final state, which will encode a permutation of the input samples that maximizes the similarity measure. A path $\pi = (\pi_0 = s_1, \pi_1, \pi_2, \ldots, \pi_n)$ is called *valid* if all the actions within its *action configuration* are distinct and each sample from the sequence $\mathcal{S}eq_\pi$ is more similar to the sample that immediately follows it in the ordered sequence than to any other sample. The state and action spaces, as well as the transition function are similar to those defined in Section 3.2.1. To obtain the optimum configuration, the reward function is defined as follows (Formula (4.1)):

$$r(\pi_k) = \begin{cases} 0 & \textit{if} \;\; k = 1 \\ -\infty & \pi \textit{ is not valid} \\ \mathrm{sim}(S_{a_{\pi_{k-1}}}, S_{a_{\pi_{k-2}}}) & \textit{otherwise} \end{cases} \tag{4.1}$$

where by $r(\pi_k)$ we denote the reward received by the agent in state π_k, after its history in the environment is $\pi = (\pi_0 = s_1, \pi_1, \pi_2, \ldots, \pi_{k-1})$, $k \in \{1, 2, \ldots, n\}$ and $a_\pi = (a_{\pi_0} a_{\pi_1} \ldots a_{\pi_{n-1}})$ is a possible action configuration.

The agent receives a negative reward on paths that are not valid, therefore it will learn to explore only valid paths. Considering the reward defined in Formula (4.1), it can be shown that the agent is trained to find a valid path that maximizes the overall similarity of the associated ordering.

4.2.2 Experimental Evaluation and Results

In this section we aim at experimentally evaluating our RL based approach for solving the TO problem on several real data sets. Some of these are time series (the correct ordering is known), thus allowing us to validate our method. Other two data sets contain samples extracted from cancer patients.

Yeast cells affected by environmental changes [GSK+00]						
Data set	**RL recovered ordering (S)**	*SMD* **(S)**	**Comp. time (sec.)**	**Ordering recovered in literature (S')**	**SMD (S')**	**Imprv.**
Heat shock	1, 2, 3, 4, 5, 6, 7, 8	0	< 2	1, 8, 7, 6, 5, 4, 3, 2 [GBJ08]	2	Yes
DTT exposure	1, 2, 3, 4, 5, 6, 7, 8	0	< 2	1, 2, 3, 4, 5, 6, 7, 8 [GBJ08]	0	Same
Amino acid starvation	1, 2, 3, 4, 5	0	< 2	1, 2, 3, 4, 5 [GBJ08]	0	Same
Nitrogen depletion	4, 3, 2, 1, 5 6, 7, 8, 9, 10	2	< 2	4, 3, 2, 1, 5, 6, 7, 8, 9, 10 [GBJ08]	2	Same
Diauxic shift	1, 2, 3, 4, 5, 6, 7	0	< 2	1, 2, 3, 4, 5, 6, 7 [GBJ08]	0	Same
α factor-based synchronization of the *Saccharomyces cerevisiae* yeast cells [SSZ+98]						
	1, 2, 3, 4, 5, 6, 7, 8, 9, 17, 14, 15, 16, 18, 10, 11, 12, 13	5	~ 5	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 18, 17, 16, 15 14, 13, 12, 11 [SSZ+98]	2	No
Response of human cells to infection by *Listeria monocytogenes* [BVBT02]						
Wild type 1	1, 2, 3, 4, 5, 6	0	< 2	1, 2, 3, 4, 5, 6 [GBJ08]	0	Same
Wild type 2	1, 2, 3, 4, 5, 6	0	< 2	3, 2, 1, 5, 4, 6[GBJ08]	4	Yes
Mutant 1	1, 4, 2, 3, 5, 6	2	< 2	1, 3, 2, 6, 5, 4 [GBJ08]	4	Yes
Mutant 2	1, 2, 3, 4, 5, 6	0	< 2	1, 2, 3, 4, 6, 5 [GBJ08]	2	Yes

Table 4.1: Presentation of the results obtained by our RL based temporal ordering algorithm for the time series data sets. For each data set, we present the solution obtained by our RL based temporal ordering algorithm, the evaluation measure SMD of the ordering, the computational time (in seconds), as well as other orderings obtained in the literature for the same data sets and their corresponding evaluation measures. The last column of the table (Improvement) specifies whether our method leads to better solutions (in terms of correct known ordering, or of lower values of the evaluation measure), compared to those that have already been reported in the literature.

4.2.2.1 Time Series Gene Expression Data

To test our method on data with known time orderings, we used several time series data sets. A time series data set is a collection of data resulted from a specific type of biological experiment: samples of tissues are extracted from the same individual at known points in time, during the progression of the biological process. The data sets used in these experiments are yeast and human time series.

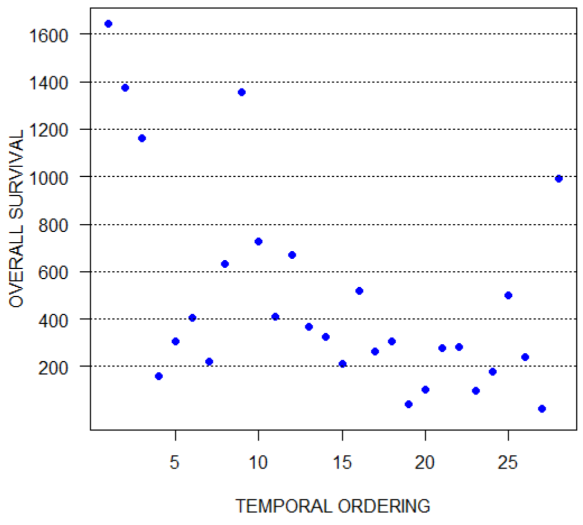
In order to be able to compare our results with other results presented in the literature for the same data sets, as well as in order to quantify the performance of our RL based algorithm, we introduce an evaluation measure which assesses the quality of a solution (ordering) obtained for a data set, with regard to the correct, known ordering. We define a measure, SMD (*Samples Misplacement Degree*), which, in our view, expresses the misplacement degree of samples in a given ordering (solution). Through its definition, the SMD measure penalizes solutions in which samples are not in the correct position in the ordering, with respect to neighboring samples.

The agent was trained using the path model Q-learning based approach, with the intelligent action selection mechanism used with $\epsilon = 0.8$ and the number of training episodes is 13000. The solutions reported in each case, after the training of the TO agent was completed are the optimal valid temporal orderings.

For each of the considered time series data sets, Table 4.1 [CBC13] presents the results. We mention that for each of the ten data sets, the correct orderings are the ones starting with the first sample (corresponding to the sample extracted at the first point in time) and increasing consecutively up to the sample which was acquired last. It can be observed that our algorithm obtained the correct orderings for *seven* out of the ten data sets. Regarding the computational time, for the smaller data sets (those containing less than, or exactly 10 samples), our RL based algorithm obtained the solutions within very short amounts of time, less than 2 seconds, on a PC at 3 GHz with 4 GB of RAM, in all nine cases (Table 4.1 [CBC13]). The computational time of our algorithm for the data set composed of 18 samples [SSZ+98] was low as well - approximately 5 seconds.

4.2.2.2 Cancer Expression Data

We also tested our RL based algorithm on a cancer gene expression data set [NMB+03], consisting of high-grade glioma samples: 28 glioblastomas and 22 anaplastic oligodendrogliomas. For each sample



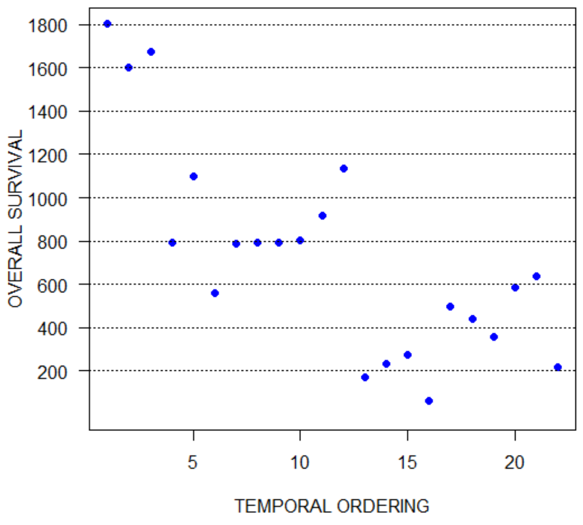


Figure 4.3: Recovered temporal orderings and survival times for the high-grade glioma data set. The figure on the left corresponds to the glioblastomas data set, while the one on the right illustrates the results for the anaplastic oligodendroglioma data set.

of these two subsets, we are given the gene expression levels for 12625 genes and the survival time following the initial diagnosis [NMB+03]. For each of the two subsets we applied the pre-processing step. Following this step, the dimensionality of the input data was significantly reduced: the input vectors for glioblastoma reached a dimensionality of 28 features (genes), while those corresponding to anaplastic oligodendrogliomas were limited to 41 features.

The TO agent was trained using the intelligent look-ahead action selection mechanism with $\epsilon = 0.8$, along $3 \cdot 10^5$ training episodes. Figure 4.3 [CBC13] presents the obtained solutions and indicates the correlation between the orderings and the survival time of the patients. The orderings are illustrated so that the first sample is always on the left and the last one on the right. It can be observed that in both cases the recovered orderings are, up to a certain extent, well correlated with the overall survival: the samples in the right half of each graph belong to patients whose survival times are lower, while the ones in the left half belong to patients having higher survival times.

4.2.3 Variations of the RL based approach

Using one of the yeast time series data sets presented in Subsection 4.2.2.1 we experimentally evaluated two more Q-learning algorithms, which use eligibility traces (Section 3.1.3): $Q(\lambda)$ [Wat89] and *naive* $Q(\lambda)$ [SB98] and two different action selection policies (*one step look-ahead procedure* - Section 3.2.4 and *softmax* action selection policy - Section 3.1.2).

The algorithms are compared by examining the accuracy of the recovered orderings, by the number of epochs they need to achieve convergence and by the computational time. The *traditional Q-learning algorithm*, with no eligibility traces, proves a very good performance obtaining the optimal solution within very short amounts of time (less than 2 seconds), when using the ϵ-Greedy based look-ahead procedure. The softmax action selection policy also leads to the correct ordering, but in this case the convergence is slower. As soon as eligibility traces are introduced, the behaviour of the Q-learning algorithm changes radically: for certain values of the policy parameters it does not converge at all, while for other values it retrieves the correct ordering, but within greater amounts of time. A possible explanation for this behaviour would be the fact that in our representation of the environment the full set of states is never completely known and therefore eligibility traces can only be updated for a known subset of states. This leads us to the conclusion that, for the *TO* problem, Q-learning with no eligibility traces is more appropriate.

4.2.4 Discussion

In this subchapter we have tackled the biological TO problem by adapting the RL based path finding model (see Section 3.2.1). To experimentally evaluate our approach, first we selected a series of data sets that have already been used in the literature [GBJ08, MLK03]. The results that we obtained are comparable and in some cases even better than the results that were reported, so far. Secondly, we evaluated our RL based method on two cancer gene expression data sets: one composed of 28 glioblastomas and the second containing 22 anaplastic oligodendrogliomas [NMB$^+$03]. Based on an intuitive correlation between the advancement of the disease and overall survival time of the patients we have chosen the survival time as a measure of validation for the obtained temporal orderings. Although for the anaplastic oligodendrogliomas data set this correlation is stronger, we may state that in both cases, the retrieved orderings are, up to a certain extent, well correlated with the survival time.

The solution to the temporal ordering problem can be used for a dual purpose. On the one hand a temporal ordering of data associated to a biological process could offer new insights into the development of the this process. On the other hand, our approach could also be used to find the correct time points of some given new samples in a pre-ordered data set. One application for this is within the field of cancer research: assuming an ordered set of cancer patients, together with their overall survival time predictions are given, when new patients, with yet unknown survival times are added to the data set, a temporal ordering of the new set (including the new patients) could reveal important information regarding their life expectancies.

As disadvantages of our approach, we mention that the only way to reduce the noise is the preprocessing step, that uses a priori knowledge on the problem. Another drawback may be the fact that a large number of training episodes is needed for large problems to obtain accurate results, but, as the experimental results have shown, good local search mechanisms may be successfully used to speed up the convergence process. We think that the direction of using RL techniques in solving the TO problem is worth being studied and further improvements can lead to more valuable results.

4.3 A Reinforcement Learning based Software Framework

In this subchapter we present a programming interface for solving optimization problems using RL techniques. The software framework was introduced in the original papers [CCB11a, CCB11b]. It was mainly developed to offer a simple and quick way to experimentally evaluate the RL based models proposed in Subchapter 3.2. Nevertheless, this framework was designed to be generic enough to be used for developing applications tackling general combinatorial optimization problems.

We propose an application programming interface (API) that allows to simply develop applications for solving combinatorial optimization problems using RL techniques. In the framework that we propose we make an abstraction of the way the optimization problem to be solved is modeled as a reinforcement learning task. This is the major advantage of our RL interface proposal: the RL algorithm is defined independent of the way the environment, states and actions are defined. Consequently, a user aiming to solve a specific optimization problem only has to define his specific state and action spaces, transition and reward functions, which are all quite simple tasks, as the abstract general versions of these RL components are already defined.

The interface is realized in JDK (Java Development Kit) 1.6 and has four basic modules: agent, environment, RL, and simulation. As in a general agent based system [Wei99], the *agent* is the entity that interacts with the *environment*, that receives perceptions and selects *actions*. The agent learns using RL to achieve its goal, i.e. to find an optimal solution of the corresponding optimization problem. Generally, the inputs of the agent are perceptions about the environment (in our case states from the environment), the outputs are actions and the environment offers rewards after interacting with it. The interaction between the agent and the environment is controlled by a *simulation* entity.

Figure 4.4 [CCB11a] shows a simplified UML diagram [(OM13] of the interface, illustrating the core of the RL interface. It is important to mention that all the classes provided by the interface remain unchanged in all applications for solving combinatorial optimization problems using reinforcement learning.

We have experimentally evaluated our RL framework, by applying it in order to obtain solutions to three optimization problems in bioinformatics, modeled as RL tasks: the biological TO problem (Section 4.2.2), DNA fragment assembly (Section 3.3.2) and protein tertiary structure prediction (Section 3.4.2). The RL framework can simply be used to develop applications for solving these

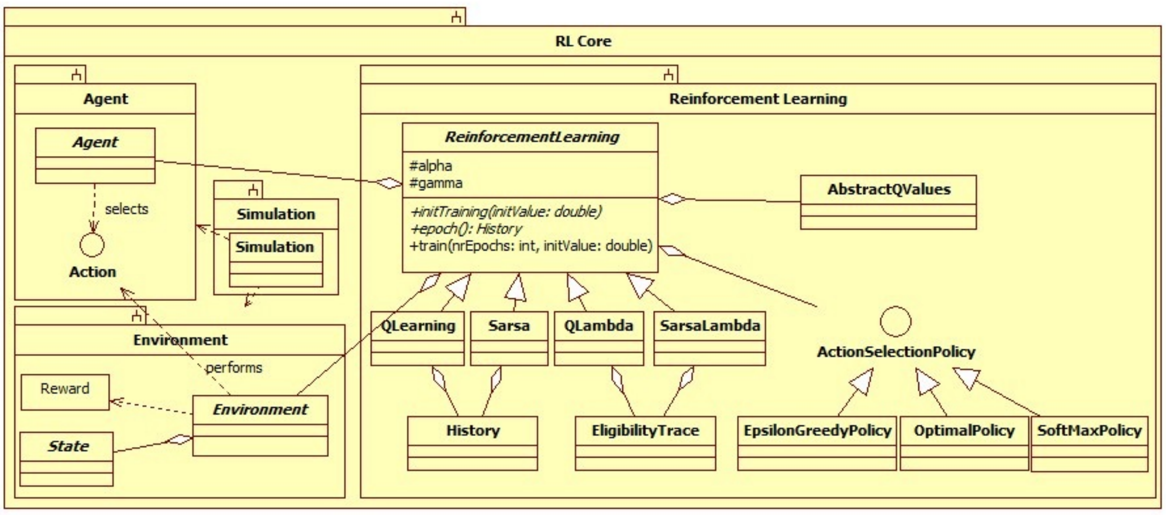


Figure 4.4: The UML diagram of the reinforcement learning based programming interface.

problems. After selecting conceptual models for the states and actions spaces and after deciding which RL algorithm and what action selection policy to use, all the user has to do is to implement new classes, which inherit or extend the interfaces or abstract classes defined in the interface.

4.4 Conclusions and Further Work

We have introduced in this chapter two different approaches for the biological temporal ordering problem: the first one proposes a methodology for obtaining temporal orderings from CNA data sets and the second uses a RL model proposed in a previous chapter (Section 3.2.1). We have also proposed a new programming interface for solving optimization problems using RL techniques, which is applied to find solutions to optimization problems in bioinformatics. The methods presented in this chapter are original works published in [CBC13, Boc12a, BCG$^+$12, BCG$^+$13, CCB11a, CCB11b].

The first approach proposes a new methodology which enables the application of a previously proposed solution for gene expression data [GBJ08] to a colorectal cancer data set, containing CNA data. Several chromosome-related measures and certain filters targeting significant portions of chromosomes are defined. Experiments are made on a data set of patients affected by colorectal cancer, at different progression stages. As for future development of this approach, we will address in deeper detail the issue of noise, we will extend its evaluation on different real life CNA data sets and we will investigate how new chromosome-related measures could influence the results.

The second method that we propose approaches the TO problem by applying a path finding RL based technique. For the experimental evaluation we use several gene expression data sets (human and yeast time series, as well as cancer expression data). Several Q-learning algorithms have been applied, using different action selection mechanisms, types of eligibility traces and various parameter settings. The good performance of the RL based model leads us to the conclusion that such machine learning models are able detect certain patterns in the input data that vary over time. We will investigate possible improvements of the RL based model for the TO problem by: using different reinforcement functions; adding various local search mechanisms; using function approximation methods to approximate the Q-values, for the cases when the state space becomes very large; considering a decreasing ϵ-Greedy strategy for the action selection mechanism. An extension of the RL based model for the TO to a distributed RL approach (Section 3.2.3) will be also considered.

Regarding the RL based software framework introduced in this chapter, we remark that it is general and it was designed to facilitate research in the direction of solving combinatorial optimization problems using RL techniques. Further work will be done in order to investigate other conceptual models for the states and actions spaces within the RL scenario of solving the bioinformatics problems that we have already approached. Also, we plan to extend the evaluation of the proposed framework for other combinatorial optimization problems.

Conclusions

The primary objectives of all research in modern biosciences is to understand the functioning of organisms, the cellular processes and metabolic pathways to the purpose of acknowledging why and how malfunctions of such processes occur. A first step in achieving this is the analysis and mining of the huge amount of collected biological and medical information. To this end, computer science offers the necessary methods, tools and algorithms. Bioinformatics has thus proven its considerable importance as a new discipline in the post-genomic era.

The research in this thesis was aimed to find solutions to several challenging problems in bioinformatics by using machine learning based models. We have particularly focused on two primary research directions. The first one is the application of *relational association rules learning* to solve classification problems in bioinformatics, which was used to find solutions to the problem of *predicting promoter sequences* in DNA molecules. The second direction refers to applying *reinforcement learning* based techniques in order to solve NP-complete combinatorial optimization problems in bioinformatics. The proposed reinforcement learning based models were applied to three important problems: *DNA fragment assembly*, *protein tertiary structure prediction* and *temporal ordering of biological samples*. In addition to these two primary research directions we also presented a novel methodology targeting a specific problem in bioinformatics and a particular type of biological data, which was developed in collaboration with the BIMIB research group from University Milano-Bicocca. Finally, we presented our original contributions towards the development of software systems by introducing a programming interface for solving optimization problems using reinforcement learning techniques.

The good performance of the classification model based on relational association rules discovery for promoter sequences prediction that we introduced leads us to the conclusion that machine learning models and data mining techniques are significant soft computing tools, able to detect and recognize patterns in biological data which are hard to be identified using conventional computational techniques.

We proposed three general reinforcement learning based models for a specific type of combinatorial optimization problems which can be regarded as generalized permutation finding problems. These models were properly modified, adapted and applied to three problems in bioinformatics. Experimental evaluations were conducted on real-life data sets from the domain of the considered problems and the obtained results proved good performances, thus demonstrating the potential of our proposals.

Comparisons with similar approaches from the literature were provided for all the models that we proposed. In many cases, our original approaches proved to outperform similar methods, thus emphasizing the efficiency of our models. Furthermore, for all cases in which different models (that are based on the same idea) are proposed, we offer comparisons and analyses of all these models.

The reinforcement learning based software framework introduced in this thesis was designed to facilitate research in the direction of solving combinatorial optimization problems using reinforcement learning techniques. Its generality allows simple development of applications for solving optimization problems using reinforcement learning. We used this framework to develop applications for solving all the three bioinformatics problems that we approached using reinforcement learning based models.

Concerning future research directions, we intend to investigate possible improvements to the proposed approaches, to further extend their evaluation using different data sets from the domain of the examined bioinformatics problems. We will also consider applying the fuzzy versions of the proposed approaches (where possible) and focus on hybridizing our models by combining them with other machine learning based techniques. Moreover, we will tackle new significant problems in bioinformatics, either by using the already proposed models, adapted and modified, or by introducing new ones.

Keywords

- bioinformatics
- computational biology
- machine learning
- promoter sequences prediction
- association rule mining
- reinforcement learning
- Q-learning
- DNA fragment assembly
- protein structure prediction
- temporal ordering
- copy number alterations
- gene expression
- microarray data
- combinatorial optimization
- software framework

Bibliography

[AAdFG99] E. Angeleri, B. Apolloni, D. de Falco, and L. Grandi. DNA Fragment Assembly Using Neural Prediction Techniques. *International Journal of Neural Systems*, 9(6):523–544, 1999.

[ACB$^+$10] C.S. Attolini, Y.K. Cheng, R. Beroukhim, G. Getz, O. Abdel-Wahab, R.L. Levine, and F. Michor. A mathematical framework to determine the temporal sequence of somatic genetic events in cancer. *Proc Natl Acad Sci USA*, 107:17604–17609, 2010.

[AHO03] C. Arima, T. Hanai, and M. Okamoto. Gene Expression Analysis Using Fuzzy K-Means Clustering. *Genome Informatics*, 14:334–335, 2003.

[Anf73] C. B. Anfinsen. Principles that govern the folding of protein chains. *Science*, 181:223–230, 1973.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[ASD$^+$10] H. Ashktorab, A.A. Schäffer, M. Daremipouran, D.T. Smoot, E. Lee, and H. Brim. Distinct genetic alterations in colorectal cancer. *PLoS ONE*, 5(1):e8879, 2010.

[BCC11a] Maria Iuliana Bocicor, Gabriela Czibula, and Istvan Gergely Czibula. A distributed Q-learning approach to fragment assembly. *Studies in Informatics and Control*, 20(3):221–232, 2011.

[BCC11b] Maria Iuliana Bocicor, Gabriela Czibula, and Istvan Gergely Czibula. A reinforcement learning approach for solving the fragment assembly problem. In *In Proceedings of the 3th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC '11)*, pages 191–198. IEEE Computer Society, 2011.

[BCG$^+$12] Iuliana M. Bocicor, Giulio Caravagna, Alex Graudenzi, Claudia Cava, Giancarlo Mauri, and Marco Antoniotti. Ordering Copy Number Alteration Data to Analyze Colorectal Cancer Progression. *EMBnet.journal*, 18(Suppl. B (NETTAB 2012)):84–86, 2012.

[BCG$^+$13] Iuliana M. Bocicor, Giulio Caravagna, Alex Graudenzi, Claudia Cava, Giancarlo Mauri, and Marco Antoniotti. Reconstructing Colorectal Cancer Progression through Copy Number Alteration Data. *Complex Systems Models in Biology and Medicine: Generic Properties and Applications*, 2013. (Will be submitted).

[BJK$^+$05a] N. Beerenwinkel, Rahnenfuhrer J., R. Kaiser, D. Hoffmann, J. Selbig, and T. Lengauer. Learning multiple evolutionary pathways from cross-sectional data. *J Comput Biol*, 12(6):584–598, 2005.

[BJK$^+$05b] N. Beerenwinkel, Rahnenfuhrer J., R. Kaiser, D. Hoffmann, J. Selbig, and T. Lengauer. Mtreemix: a software package for learning and using mixture models of mutagenetic trees. *Bioinformatics*, 21(9):2106–2207, 2005.

[BL98] B. Berger and T. Leighton. Protein folding in hp model is np-complete. *Journal of Computational Biology*, 5:27–40, 1998.

[Boc10a] Maria Iuliana Bocicor. Algoritmi evolutivi aplicai n chemoterapie. In *National Symposum "Interferente", 1st Edition*, pages 145–148. Ceconi Baia Mare, 2010.

[Boc10b] Maria Iuliana Bocicor. Bioinformatica si aplicatiile ei. *Scoala Maramureseana*, (41-45):222–223, 2010.

[Boc11a] Maria Iuliana Bocicor. Invatarea automata pentru identificarea regiunilor promotor in adn. In *National Symposum "Interferente", 3rd Edition*, pages 68–70. Universitatea de Nord Baia Mare, 2011.

[Boc11b] Maria Iuliana Bocicor. Modele pentru problema plierii proteinei. In *National Symposum "Interferente", 2nd Edition*, pages 191–193. Universitatea de Nord Baia Mare, 2011.

[Boc12a] Iuliana M. Bocicor. A Study on Using Reinforcement Learning for Temporal Ordering of Biological Samples. *Studia Universitatis "Babes-Bolyai" Informatica*, LVII(4):62–74, 2012.

[Boc12b] Maria Iuliana Bocicor. Experiments on promoter sequences prediction using association rules. In *In Proceedings "Zilele Academice Clujene 2012, Departamentul de Informatica"*, pages 32–35. Presa Universitara Clujeana, 2012.

[Boc12c] Maria Iuliana Bocicor. A study on using association rules for predicting promoter sequences. *Studia Universitatis "Babes-Bolyai", Informatica*, LVII(2):32–42, 2012.

[BPSS01] A. Brazma, H. Parkinson, T. Schlitt, and M. Shojatalab. A quick introduction to elements of biology - cells, molecules, genes, functional genomics, microarrays. `http://www.ebi.ac.uk/microarray/biology_intro.html`, 2001. Accessed: 30 August 2010.

[BS06] E. Bindewald and B.A. SHAPIRO. RNA secondary structure prediction from sequence alignments using a network of k-nearest neighbor classifiers. *RNA*, 12:342–352, 2006.

[BVBT02] David N. Baldwin, Veena Vanchinathan, Patrick O. Brown, and Julie A. Theriot. A gene-expression program reflecting the innate immune response of cultured intestinal epithelial cells to infection by Listeria monocytogenes. *Genome Biology*, 4(1):4241–4257, 2002.

[BWTB08] E. Bolton, Y. Wang, P.A. Thiessen, and S.H. Bryant. PubChem: Integrated Platform of Small Molecules and Biological Activities. In *Annual Reports in Computational Chemistry*, volume 4, chapter 12. American Chemical Society, Washington DC, 2008.

[CBC11a] Gabriela Czibula, Maria Iuliana Bocicor, and Istvan Gergely Czibula. A distributed reinforcement learning approach for solving optimization problems. In *In Proceedings of the 5th International Conference on Communications and Information Technology (CIT '11)*, pages 25–30, 2011.

[CBC11b] Gabriela Czibula, Maria Iuliana Bocicor, and Istvan Gergely Czibula. An experiment on protein structure prediction using reinforcement learning. *Studia Universitatis "Babes-Bolyai", Informatica*, LVI(1):25–34, 2011.

[CBC11c] Gabriela Czibula, Maria Iuliana Bocicor, and Istvan Gergely Czibula. A reinforcement learning model for solving the folding problem. *International Journal of Computer Technology and Applications*, 2(1):171–182, 2011.

[CBC11d] Gabriela Czibula, Maria Iuliana Bocicor, and Istvan Gergely Czibula. Solving the protein folding problem using a distributed Q-learning approach. *International Journal of Computers*, 5(3):404–413, 2011.

[CBC12] Gabriela Czibula, Maria Iuliana Bocicor, and Istvan Gergely Czibula. Promoter sequences prediction using relational association rule mining. *Evolutionary Bioinformatics*, 8:181–196, 2012.

[CBC13] Gabriela Czibula, Maria Iuliana Bocicor, and Istvan Gergely Czibula. Temporal ordering of cancer microarray data through a reinforcement learning based approach. *PloS ONE*, 8(4):e60883, 2013.

[CBL$^+$12] Y.K. Cheng, R. Beroukhim, R.L. Levine, I.K. Mellinghoff, E.K. Holland, and F. Michor. A Mathematical Methodology for Determining the Temporal Order of Pathway Alterations Arising during Gliomagenesis. *PLOS Computational Biology*, 8(1):1–15, 2012.

[CCB11a] Istvan Gergely Czibula, Gabriela Czibula, and Maria Iuliana Bocicor. A Software Framework for Solving Combinatorial Optimization Tasks. *Studia Universitatis "Babes-Bolyai", Informatica*, Special Issue, LVI(3):3–8, 2011.

[CCB11b] Istvan Gergely Czibula, Gabriela Czibula, and Maria Iuliana Bocicor. A reinforcement learning based framework for solving optimization problems. In *In Post proceedings of Knowledge Egineering Principles and Techniques*, pages 235–246. Presa Universitara Clujeana, 2011.

[CCB13] Gabriela Czibula, Istvan Gergely Czibula, and Maria Iuliana Bocicor. A Comparison of Reinforcement Learning Based Models for the DNA Fragment Assembly Problem. *Studia Universitatis "Babes-Bolyai" Informatica*, LVIII(2):90–102, 2013.

[Chi10] C. Chira. Hill-Climbing Search in Evolutionary Models for Protein Folding Simulations. *Studia*, LV:29–40, 2010.

[CK91] David Chapman and Leslie Pack Kaelbling. Input generalization in delayed reinforcement learning: an algorithm and performance comparisons. In *Proceedings of the 12th international joint conference on Artificial intelligence - Volume 2*, pages 726–731, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc.

[CL05] Shyi-ming Chen and Chung-hui Lin. Multiple dna sequence alignment based on genetic algorithms and divide-and-conquer techniques. *International Journal of Applied Science and Engineering*, 3:89–100, 2005.

[Coo11] William Cook. Concorde TSP solver, 2011. http://www.tsp.gatech.edu/concorde.html.

[CPCC06] Y. Chen, Y. Pan, L. Chen, and J. Chen. Partitioned optimization algorithms for multiple sequence alignment. *Proceedings of the 20th International Conference on Advanced Information Networking and Applications*, pages 618–622, 2006.

[CS94] Mark W. Craven and Jude W. Shavlik. Machine learning approaches to gene recognition. *IEEE Intelligent Systems*, 9(2):2–10, 1994.

[CSS99] William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.

[CSTM06] A. Câmpan, G. Serban, T.M. Truta, and A. Marcus. An Algorithm for the Discovery of Arbitrary Length Ordinal Association Rules. In *DMIN*, pages 107–113, 2006.

[DD01] Chris H. Q. Ding and Inna Dubchak. Multi-class Protein Fold Recognition Using Support Vector Machines and Neural Networks. *Bioinformatics*, 17:349–358, 2001.

[Dil85] K.A. Dill. Theory for the folding and stability of globular proteins. *Biochemistry*, 24(6):1501–1509, 1985.

[DJK$^+$99] Richard Desper, Feng Jiang, Olli P. Kallioniemi, Holger Moch, Christos H. Papadimitriou, and Alejandro A. Schaffer. Inferring Tree Models for Oncogenesis from Comparative Genome Hybridization Data. *Journal of Computational Biology*, 6(1):37–51, 1999.

[DJK$^+$00] Richard Desper, Feng Jiang, Olli P. Kallioniemi, Holger Moch, Christos H. Papadimitriou, and Alejandro A. Schaffer. Distance-based reconstruction of tree models for oncogenesis. *J Comput Biol*, 7(6):789–803, 2000.

[EB96] M.L. Engle and C. Burks. Artificially generated data sets for testing dna fragment assembly algorithms. *Genomics*, 16(1):286–288, 1996.

[FA10] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[Fog05] G.B. Fogel. Gene expression analysis using methods of computational intelligence. *Pharmaceutical Discovery*, 5:12–18, 2005.

[Fra07] S.A. Frank. *Dynamics of Cancer.* Princeton University Press, 2007.

[FV90] E.R. Fearon and B. Volgestein. A genetic model for colorectal tumorigenesis. *Cell*, 61:759–767, 1990.

[GBF$^+$07] Thiago Gonzaga, Cristina Bentes, Ricardo Farias, Maria Clicia Castro, and Ana Cristina Garcia. Using distributed-shared memory mechanisms for agents communication in a distributed system. In *Proceedings of the Seventh International Conference on Intelligent Systems Design and Applications*, pages 39–46, Washington, DC, USA, 2007. IEEE Computer Society.

[GBHB09] M. Gerstung, M. Baudis, Moch. H, and N. Beerenwinkel. Quantifying cancer progression with conjunctive Bayesian networks. *Bioinformatics*, 25(21):2809–2815, 2009.

[GBJ08] Anupam Gupta and Ziv Bar-Joseph. Extracting Dynamics from Static cancer Expression Data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 5(2):172–182, June 2008.

[GEL$^+$11] Moritz Gerstung, Nicholas Eriksson, Jimmy Lin, Bert Volgestein, and Niko Beerenwinkel. The Temporal Order of Genetic and Pathway Alterations in Tumorigenesis. *PLoS ONE*, 6(11):1–9, 2011.

[GP08] Ashish Ghosh and Bijnan Parai. Protein secondary structure prediction using distance based classifiers. *International Journal of Approximate Reasoning*, 47:37–44, 2008.

[GSK$^+$00] Audrey P. Gasch, Paul T. Spellman, Camilla M. Kao, Orna Carmel-Harel, Michael B. Eisen, Gisela Storz, David Botstein, and Patrick O. Brown. Genomic Expression Programs in the Response of Yeast Cells to Environmental Changes. *Molecular Biology of the Cell*, 11(12):4241–4257, 2000.

[HCP$^+$00] K.B. Hwang, D.Y. Cho, Wook Park, Kim S.W., Zhang S.D., and B.Y. Applying machine learning techniques to analysis of gene expression data: Cancer diagnosis. *In: Proc. 1st Conf. on Critical Assessment of Microarray Data Analysis*, 2000.

[HH96] Mance E. Harmon and Stephanie S. Harmon. Reinforcement learning: A tutorial, 1996.

[HHL06] M. Hjelm, M. Hoglund, and J. Lagergren. New probabilistic network models and algorithms for oncogenesis. *Journal of Computtational Biology*, 13(4):853–865, 2006.

[HL03] C. J. Huang and W.-C. Liao. A comparative study of feature selection methods for probabilistic neural networks in cancer classification. *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*, pages 451–458, 2003.

[HMTA08] A.E. Hassanien, M.G. Milanova, Smolinski T.G., and Abraham A. Computational Intelligence in Solving Bioinformatics Problems: Reviews, Perspectives, and Challenges. *Computational Intelligence in Biomedicine and Bioinformatics Studies in Computational Intelligence*, 151:3–47, 2008.

[HYYY02] D. N. Hung, I. Yoshihara, K. Yamamori, and M. Yasunaga. A parallel hybrid genetic algorithm for multiple protein sequence alignment. *Proceedings of the 2002 Congress on Evolutionary Computation*, pages 309–314, 2002.

[JDH99] T. Jaakkola, M Diekhans, and D Haussler. Using the sher kernel method to detect remote protein homologies. *In ISMB*, pages 149–158, 1999.

[JSXW10] A. Jemal, R. Siegel, J. Xu, and E. Ward. Cancer statistics 2010. *CA Cancer J. Clin.*, 60:277–300, 2010.

[KC06] Satoko Kikuchi and Goutam Chakraborty. Heuristically Tuned GA to Solve Genome Fragment Assembly Problem. *IEEE CEC*, pages 1491–1498, 2006.

[Klo72] A.H. Klopf. Brain function and adaptive systems. A heterostatic theory. *Technical Report AFCRL-72-0164*, 1972.

[Kos07] Walter Kosters. Bioinformatics: Fragment Assembly. *IPA–Algorithms and Complexity - course*, 2007.

[KP04] Nikola Kasabov and Shaoning Pang. Transductive support vector machines and applications in bioinformatics for promoter recognition. *Neural Information Processing - Letters and Reviews*, 3(2):31–37, 2004.

[LAR$^+$10] Geer L.Y., Marchler-Bauer A., Geer R.C., Han L., He J., He S., Liu C., Shi W., and Bryant S.H. The ncbi biosystems database. *Nucleic Acids Research*, (38(Database issue)), 2010.

[LATK06] G. Luque, E. Alba Torres, and S. Khuri. Assembling DNA Fragments with a Distributed Genetic Algorithm. *Parallel Computing for Bioinformatics and Computational Biology*, pages 285–302, 2006.

[LEN02] Christina Leslie, Eleazar Eskin, and William Stafford S. Noble. The spectrum kernel: a string kernel for SVM protein classification. *Pacific Symposium on Biocomputing*, pages 564–575, 2002.

[LHJYFHB04] Wang Long-Hui, Liu Juan, Li Yan-Fu, and Zhou Huai-Bei. Predicting protein secondary structure by a support vector machine based on a new coding scheme. *Genome Informatics*, 15:181–190, 2004.

[LK04] Lishan Li and Sami Khuri. A comparison of DNA fragment assembly algorithms. In *Proceedings of the International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences*, pages 329–335. CSREA Press, 2004.

[MC03] P. Meksangsouy and N. Chaiyaratana. DNA fragment assembly using an ant colony system algorithm. In *Proceedings of CEC'03 - vol.3*, pages 1756–1763. IEEE Press, 2003.

[MGR06] J. Martin, J. F. Gibrat, and F. Rodolphe. Analysis of an optimal hidden markov model for secondary structure prediction. *BMC Structural Biology*, 6:25–45, 2006.

[Mit97] T.M. Mitchell. *Machine Learning*. New York: McGraw-Hill, 1997.

[MLK03] Paul M. Magwene, Paul Lizardi, and Junhyong Kim. Reconstructing the temporal ordering of biological samples using microarray data. *Bioinformatics*, 19(7):842–850, 2003.

[MML01] A. Marcus, J.I. Maletic, and K.I. Lin. Ordinal association rules for error identification in data sets. In *Proceedings of the tenth international conference on Information and knowledge management*, CIKM '01, pages 589–591, New York, NY, USA, 2001. ACM.

[Net12] The Cancer Genome Atlas Network. Comprehensive molecular characterization of human colon and rectal cancer. *Nature*, 487(7407):330–337, 2012.

[NMB$^+$03] Catherine L. Nutt, D. R. Mani, Rebecca A. Betensky, Pablo Tamayo, J. Gregory Cairncross, Christine Ladd, Ute Pohl, Christian Hartmann, Margaret E. McLaughlin, Tracy T. Batchelor, Peter M. Black, Andreas von Deimling, Scott L. Pomeroy, Todd R. Golub, and David N. Louis. Gene Expression-based Classification of Malignant Gliomas Correlates Better with Survival than Histological Classification. *Cancer Research*, 63(7):1602–1607, 2003.

[NT88] Qian Ning and Sejnowski Terrence. Predicting the secondary structure of globular proteins using neural network models. *J. Mol. Biol.*, 202:865–884, 1988.

[NVNM07] S. Nasser, G.L. Vert, M. Nicolescu, and A. Murray. Multiple sequence alignment using fuzzy logic. *Proceedings IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology*, pages 304–311, 2007.

[(OM13] Object Management Group (OMG). UML Resource Page. `http://uml.org/`, 2013. Accessed: 10 May 2013.

[OP09] O. Okun and H Priisalu. Dataset complexity in gene expression based cancer classification using ensembles of k-nearest neighbors. *Artificial Intelligence in Medicine*, 45(2-3):151–162, 2009.

[PE95] Anders Gorm Pedersen and Jacob Engelbrecht. Investigations of Escherichia coli Promoter Sequences With Artificial Neural Networks: New Signals Discovered Upstream of the Transcriptional Startpoint. *Proceedings of the International Conference on Intelligent Systems for Molecular Biology*, 3:292–299, 1995.

[Pev00] Pavel A. Pevzner. Computational molecular biology: An algorithmic approach. 2000.

[PFB95] Rebecca J. Parsons, Stephanie Forrest, and Christian Burks. Genetic Algorithms, Operators, and DNA Fragment Assembly. In *Machine Learning*, pages 11–33. Kluwer Academic Publishers, 1995.

[PPN09] Amiya Kumar Patel, Seema Patel, and Pradeep Kumar Naik. Binary Classification of uncharacterized uncharacterized proteins int DNA binding/non-DNA binding proteins from sequence derived features using ANN. *Digest Journal of Nanomaterials and Biostructures*, 4:775–782, 2009.

[PS94] Dayan P. and T.J. Sejnowski. Td(lambda) converges with probability 1. *Machine Learning*, 14:295–301, 1994.

[PSBM09] S. Pathare, A. Schaffer, N. Beerenwinkel, and M Mahimkar. Construction of oncogenetic tree models reveals multiple pathways of oral cancer progression. *International Journal of Cancer*, 124(12):2864–2871, 2009.

[PU98] A. Perez-Uribe. Introduction to reinforcement learning, 1998. http://lslwww.epfl.ch/∼anperez/RL/RL.html.

[RGS$^+$09] J.F. Reid, M. Gariboldi, V. Sokolova, P. Capoblanco, A. Lampis, F. Perrone, S. Signoroni, A. Costa, E. Leo, S. Pilotti, and M.A. Pierotti. Integrative Approach for Prioritizing Cancer Genes in Sporadic Colon Cancer. *Genes, Chromosomes and Cancer*, 48:953–962, 2009.

[RK03] T.K. Rasmussen and T Krink. Improved hidden markov model training for multiple sequence alignment by a particle swarm optimization-evolutionary algorithm hybrid. *BioSystems*, 72:5–17, 2003.

[SB98] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.

[SCC06] G. Serban, A. Câmpan, and I.G. Czibula. A Programming Interface for Finding Relational Association Rules. *International Journal of Computers, Communications & Control*, I(S.):439–444, 2006.

[SH05] A. Shmygelska and H.H. Hoos. An ant colony optimisation algorithm for the 2D and 3D hydrophobic polar protein folding problem. *BMC Bioinformatics*, 6, 2005.

[Spe04] C. Spearman. The proof and measurement of association between two things. *Amer. J. Psychol.**15***, pages 72–101, 1904.

[SS96] S. P. Singh and R. S. Sutton. Reinforcement Learning with Replacing Eligibility Traces. *Machine Learning*, 22:123–158, 1996.

[SS05] M. Sevaux and K. Sorensen. Permutation distance measures for memetic algorithms with population management. In *Proceedings of the The Sixth Metaheuristics International Conference*, MIC'05, 2005.

[SSZ$^+$98] Paul T. Spellman, Gavin Sherlock, Michael Q. Zhang, Vishwanath R. Iyer, Kirk Anders, Michael B. Eisen, Patrick O. Brown, David Botstein, and Bruce Futcher. Comprehensive Identification of Cell Cycleregulated Genes of the Yeast *Saccharomyces cerevisiae* by Microarray Hybridization. *Mol. Biol. Cell*, 9(12):3273–3297, 1998.

[Ste93] Evan W. Steeg. Neural networks, adaptive optimization, and RNA secondary structure prediction. In *In Artificial Intelligence and Molecular Biology*, pages 121–160, 1993.

[Thr92] Sebastian Thrun. The Role of Exploration in Learning Control. In *Handbook for Intelligent Control: Neural, Fuzzy and Adaptive Approaches.* Van Nostrand Reinhold, Florence, Kentucky, 1992.

[TMM08] T. Thalheim, D. Merkle, and M. Middendorf. Protein Folding in the HP-Model Solved With a Hybrid Population Based ACO Algorithm. *IAENG International Jurnal of Computer Science*, 35, 2008.

[TPAG11] Uma Devi Tatavarthi, Venkata Nageswara Rao Padmanbhuni, Appa Rao Allam, and Ramachandra Sridhar Gumpeny. In silico promoter prediction using grey relational analysis. *Journal of Theoretical and Applied Information Technology*, 24(2):107–112, 2011.

[TSN90] Geoffrey G. Towell, Jude W. Shavlik, and Michiel O. Noordewier. Refinement of approximate domain theories by knowledge-based artificial neural networks. *In Proceedings of the Eighth National Conference on Artificial Intelligence(AAAI-90)*, pages 861–866, 1990.

[Tuf11] Stephane Tuffery. *Data Mining and Statistics for Decision Making.* John Wiley and Sons, 2011.

[TYA08] N.T. Tung, E. Yang, and I.P Androulakis. Machine Learning Approaches in Promoter Sequence Analysis. *In Machine Learning Research Progress*, 2008.

[UM93] R. Unger and J. Moult. Genetic Algorithms for Protein Folding Simulations. *Journal of Molecular Biology*, 231:75–81, 1993.

[VFH+88] B. Vogelstein, E.R. Fearon, S.R. Hamilton, S.E. Kern, A.C. Preisinger, M. Leppert, Y. Nakamura, R. White, A.M. Smits, and J.L. Bos. Genetic alterations during colorectal-tumor development. *N. Engl. J. Med.*, 319:3526–3535, 1988.

[Wat89] C. J. C. H. Watkins. Learning from Delayed Rewards. *PhD thesis*, 1989.

[WCT06] W. Wetcharaporn, N. Chaiyaratana, and S. Tongsima. DNA Fragment Assembly by Ant Colony and Nearest Neighbour Heuristics. *ARTIFICIAL INTELLIGENCE AND SOFT COMPUTING ICAISC 2006*, pages 1008–1017, 2006.

[WD92] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

[Wei99] Gerhard Weiß. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence.* MIT Press, Cambridge, MA, USA, 1999.

[WLD03] D. Wang, N.K. Lee, and T.S. Dillon. Extraction and optimization of fuzzy protein sequences classification rules using GRBF neural networks. *Neural Information Processing - Letters and Reviews*, 1:53–57, 2003.

[XDE+03] X. Xiao, E.R. Dow, R.C. Eberhart, Z.B. Miled, and R.J. Oppelt. Gene Clustering Using Self-Organizing Maps and Particle Swarm Optimization. *In: Proc. 17th Intl. Symposium on Parallel and Distributed Processing*, 2003.

[YLGW02] Y. Yuhui, C. Lihui, A. Goh, and A. Wong. Clustering gene data via associative clustering neural network. *In: Proc. 9th Intl. Conf. on Information Processing*, pages 2228–2232, 2002.

[YV04] B-Y Yoon and P.P. Vaidyanathan. Hmm with auxiliary memory: a new tool for modeling RNA secondary structures. In *Proc. 38th Asilomar Conference on Signals, Systems, and Computers*, pages 1651–1655, 2004.

[ZCY+11] Wenyu Zhang, Jiajia Chen, Yang Yang, Yifei Tang, Jing Shang, Bairong Shen, and I. King Jordan. A practical comparison of de novo genome assembly software tools for next-generation sequencing technologies. *PLoS ONE*, 6(3):e17915, 2011.

[ZWL+09] X. Zhang, T. Wang, H. Luo, Y.J. Yang, Y. Deng, J. Tang, and M. Q. Yang. 3D Protein structure prediction with genetic tabu search algorithm. *BMC Systems Biology*, 4, 2009.