A Computational Framework for Protein-DNA Binding Discovery

WONG, Ka Chun

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong
August 2010

Thesis/Assessment Committee

Professor LEE, Kin Hong (Chair)
Professor LEUNG, Kwong Sak (Thesis Supervisor)
Professor WONG, Man Hon (Thesis Supervisor)
Professor LAU, Lap Chi (Committee Member)
Professor Junzo WATADA (External Examiner)

"Biology and computer science - life and computation - are related. I am confident that at their interface great discoveries await those who seek them."

Leonard Adleman, Scientific American Magazine 1998

THE CHINESE UNIVERSITY OF HONG KONG

Abstract

Faculty of Engineering
Department of Computer Science and Engineering

Master of Philosophy

by WONG KA CHUN

With increasing computational power, availability of databases with massive protein and DNA data, and mature data mining techniques, a framework is proposed to discover associated protein-DNA binding sequence patterns from TRANSFAC.

To further analyze the discovered sequence patterns in the huge search space, two evolutionary algorithms are proposed. In particular, the evolutionary algorithms are specially designed for multimodal optimization to avoid premature convergence and genetic drift. The one with less number of parameters (CrowdingGA-L) is selected and applied to learn the protein-DNA bindings in generalized sequence representations. Some promising results are obtained.

As a further application, CrowdingGA-L is also applied to predict protein structures on a lattice model. The experiments show that it can be applied to obtain results comparable with the other state-of-the-art algorithms, although it is a relatively simple technique.

香港中文大學

摘要

工程學院
計算機科學及工程學系
哲學碩士
黃家駿 著

結合逐漸強大的計算能力，大量的脫氧核糖核酸和蛋白質實驗資料庫及成熟的數據挖掘技術，本論文提出了一個計算框架以從 TRANSFAC 發現相關的脫氧核糖核酸和蛋白質結合的序列模式。

為了進一步在巨大的模式搜索空間分析該些序列模式，本論文提出兩個特別設計的進化算法以進行多模態優化，以避免過早收斂和遺傳漂移。其中一個參數數量較少的進化算法（CrowdingGA-L）被選擇應用以推廣該些模式，從而獲得有希望的結果。

作為進一步的應用，CrowdingGA-L 也被選擇應用在格模型對蛋白質結構的預測。實驗表明，雖然它是一個相對簡單的技術，但它卻可以得到與其他最先進的算法不相伯仲的成績。

Acknowledgements

Taking up a research project is never an easy job. It always needs many supports from supervisors and colleagues. Therefore, given this chance, I would like to thank the ones who gave me support, guidance, and encouragement.

Most importantly, I would like to acknowledge my respectable supervisors: Prof. Kwong-Sak Leung and Prof. Man-Hon Wong. They guided me to be on the right way, inspired me to find out many possibilities on doing the project. Although I could not meet their requirements sometimes, they were still very patient in teaching me. Their guidance was indispensable to the completion of this thesis.

I would also like to show my gratitude to Prof. Kin-Hong Lee, Prof. Stephen K.W. Tsui, and Prof. Chi-Kong Lau. They shared with me research knowledge and gave me some insightful suggestions. Especially, they kept me up to date in the bioinformatics research field.

Last but not least, I would like to give thanks to my colleagues: Chan Tak Ming, Lam Chi Fai, Lo Leung Yau, Ni Bing, Li Gang, Ling Kin Cheung, Tse Ching Man, Yip Chun Ting, Li Hong Jian, Wang Jin Feng, and Li Wen Ye. They are enthusiastic to answer my questions. I appreciate them very much.

Finally, I would like to thank all of them again. Without any of them, the works in this thesis cannot be carried out smoothly.

Contents

List of Figures

List of Tables

Dedicated to my parents

Chapter 1

Introduction

1.1 Motivation

Protein-DNA bindings between transcription factors (TFs) and transcription factor binding sites (TFBSs) play an essential role in transcriptional regulation. However, it is expensive and laborious to experimentally identify TF-TFBS binding sequence pairs, for example, using DNA footprinting [3] or gel electrophoresis [4]. The technology of Chromatin immunoprecipitation (ChIP) [5, 6] measures the binding of a particular TF to DNA of co-regulated genes on a genome-wide scale *in vivo*, but at low resolution. Further processing are needed to extract precise TFBSs [7]. TRANSFAC [8] is one of the largest and most representative databases for regulatory elements including TFs, TFBSs, nucleotide distribution matrices of the TFBSs, and regulated genes. The data are expertly annotated and manually curated from peer-reviewed and experimentally proved publications. Other annotation databases of TF families and binding domains are also available (e.g. PROSITE [9], Pfam [10]). It is even more difficult and time consuming to extract high-resolution 3D TF-TFBS complex structures with X-ray crystallography or Nuclear Magnetic Resonance (NMR) spectroscopic analysis. Nevertheless, the high-quality TF-TFBS binding structures provide valuable insights into verifications of putative principles of binding. The Protein Data Bank (PDB) [11] serves as a representative repository of such experimentally extracted protein-DNA (in particular TF-TFBS) complexes with high resolution at atomic levels. However, the available 3D structures are far from complete. As a result, there is strong motivation to have automatic methods, particularly, computational approaches based on existing abundant data, to provide testable candidates of novel TF domains and/or TFBS motifs with high confidence to guide and accelerate the wet-lab experiments.

1.2 Objective

To propose computational methods and apply them to provide testable candidates of novel TF-TFBS binding sequence pairs with high confidence to guide and accelerate the wet-lab experiments.

1.3 Methodology

A bioinformatics framework is proposed to discover associated TF-TFBS binding sequence patterns from TRANSFAC. To further analyze the discovered sequence patterns in the huge search space, two evolutionary algorithms are proposed. In particular, the evolutionary algorithms are specially designed for multimodal optimization to avoid premature convergence and genetic drift. The one with less number of parameters (CrowdingGA-L) is selected and applied to generalize the sequence representations. Some promising results are obtained. As a further application, CrowdingGA-L is also applied to predict protein structures on a lattice model. The experiments show that it can be applied to obtain results comparable with the other state-of-the-art algorithms, although it is a relatively simple technique.

1.4 Bioinformatics

In recent years, genome sequencing projects around the world have successfully worked out the whole genomes for different species . In 1995, the first free-living organism *Haemophilus influenzae* was sequenced by The Institute for Genomic Research[12]. In 1996, the first eukaryotic genome was completely sequenced. It was the model eukaryote species, *Saccharomyces cerevisiase*[13]. In 2000, the first plant genome, *Arabidopsis thaliana*, was also sequenced by Arabidopsis Genome Initiative[14]. Finally in 2006, with the last chromosome sequenced, the Human Genome Project (HGP) announced its completion[15]. Although the above, the story has not ended yet. Merely raw genomic sequence data is not sufficient for scientists to jump into any meaningful conclusions. Further data analysis of the genomes is required.

In particular, computational methods have been attracting increasing attention due to its high speed and low cost, comparing to wet-lab experiments. They are collectively known as bioinformatics. For instance, motif discovery [16] helps us distinguish real signals from the background sequences. Multiple sequence alignment [17] can be used to study the similarity between multiple sequences. Protein structure prediction [18] can be applied to predict the 3D tertiary structure from an amino acid sequence. Gene network inference [19] uses statistical methods to infer gene network from microarray

data. Promoter prediction [20] help us annotate the promoter region in a genome. Phylogenetic tree inference [21] can be used to study the relationship between different species. Drug scheduling [22] can help doctors solve the clinical problems in an effective manner. Although the accuracy of the methods is sometimes lower than that of wet-lab experiments, they can still serve as useful preprocessing tools to significantly narrow the search space. Thus putative candidates can be selected for further validation by wet-lab experiments. Time and money are saved.

1.5 Computational Methods

1.5.1 Evolutionary Algorithms

Evolutionary algorithm builds a bridge between computer science and natural science [23]. Instead of artificial creation, evolutionary algorithm emphasizes on learning from the nature. Nature rules are applied or modeled to build brand-new computational techniques, which can be well adapted and integrated into different contexts. For instance, inspired from the Darwin's evolutionary theory, John Holland has proposed a computational technique called genetic algorithm which resembles the evolutionary process for natural selection. In a typical run, it simulates the natural mechanism of a group of individuals (called population). The individuals perform crossovers with each other to form offspring, who probabilistically undergo mutations. After that, it comes to survival selection. Fitter offspring will be retained and kept to the next generation. The offspring become parents. They, again, crossover with each other to form offspring. Iteratively, it is expected the population become fitter and fitter. Optimization can thus be achieved.

Since genetic algorithm was proposed by John Holland [24] in the early 1970s, evolutionary algorithm has emerged as a popular research field. Researchers from various scientific and engineering disciplines have been digging into this field and exploring the power of evolutionary algorithms. Many international conferences specialized for evolutionary algorithms have been created like ACM GECCO, IEEE CEC, EvoStar, and PPSN... With such a diverse base of researchers, many applications have been successfully proposed in the past twenty years. For example, mechanical design [25], electromagnetic optimization [26], environmental protection [27], finance [28], musical orchestration [29], pipe routing [30], and nuclear reactor core design [31]. In particular, its function optimization capability was highlighted [32] because of its high adaptability to different function landscapes, to which we cannot apply traditional optimization techniques.

The design of evolutionary algorithms draws inspiration from the nature. They simulate the natural mechanism and are closer to the nature. It is intuitive that they should be among the best methods in bioinformatics to decipher the nature . Thus evolutionary algorithms are heavily analysed and involved in the framework proposed in this thesis.

1.5.2 Data Mining for TF-TFBS bindings

The first attempt of computational methods related to TF-TFBS bindings was to discover the motifs of TF domains and TFBSs separately. Many data mining approaches have also been proposed [33]. Researchers employ and transfer additional detailed information such as base compositions, structures, thermodynamic properties [34, 35] as well as expressions [36], into sophisticated features to fit into certain data mining techniques. Although some approaches may provide interpretable rules, most of them have stringent data requirements which cannot be obtained trivially. Existing data beyond sequences are also insufficient and limited for practitioners. These methods usually extract complicated features rather than working on interpretable data directly. Many data mining techniques, such as neural networks, support vector machines (SVM) [37], and regressions [33], may generate rules which are not trivial to interpret. Furthermore, many data mining approaches are based on specific families or particular datasets, where the generality of the results are limited. On the other hand, sequences serve as the most handy primary data which carry important information for protein-DNA bindings [38]. It is desirable to make use of the large-scale and comprehensive sequence data to mine explicit and interpretable TF-TFBS binding rules. A framework for discovering the binding information is thus proposed in this thesis.

Chapter 2

Background

2.1 Gene Transcription

DNA makes RNA, RNA makes Protein is the central dogma of molecular biology. The first process is called **Transcription**, whereas the second process is called **Translation**. For each gene being expressed, there is probably a promoter region upstream of it. When several transcription factors (TFs) bind on the transcription factor binding sites (TFBSs) on a promoter region, it will direct an RNA polymerase to the correct transcriptional start site and transcript the target gene into RNA. The above gene regulatory process is called **Transcription**, whereas the elements involved are called **Transcription regulatory elements**. There are also some enhancer and silencer regions around the promoter region which can enhance and hinder the transcription process at a certain level respectively, as shown in Figure 2.1.

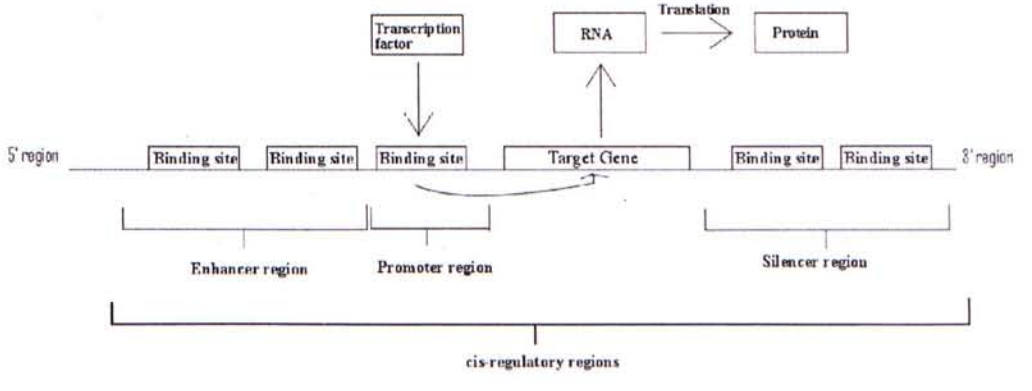


FIGURE 2.1: An example of cis- and trans-regulatory elements for a gene

2.1.1 Protein-DNA Binding

Protein-DNA binding plays a central role in genetic activities such as transcription, packaging, rearrangement, and replication [39, 40]. Therefore it is very important to identify and understand the protein-DNA bindings as the basis for further deciphering biological systems. We focus on protein-DNA bindings between Transcription Factors (TFs) and Transcription Factor Binding Sites (TFBSs), which are the primary regulatory activities with abundant data support. TFs bind in a sequence-specific manner to TFBSs to regulate gene transcription (gene expression). The DNA binding domain(s) of a TF can recognize and bind to a collection of similar TFBSs, from which a conserved pattern called motif can be obtained. TFBSs, the nucleotide fragments bound by TFs, are usually short (usually about 5 - 20 bp) in the cis-regulatory/intergenic regions, and can assume very different locations from the transcription start site (TSS).

It is expensive and laborious to experimentally identify TF-TFBS binding sequence pairs, for example, using DNA footprinting [3] or gel electrophoresis [4]. The technology of Chromatin immunoprecipitation (ChIP) [5, 6] measures the binding of a particular TF to DNA of co-regulated genes on a genome-wide scale *in vivo*, but at low resolution. Further processing are needed to extract precise TFBSs [7]. TRANSFAC [8] is one of the largest and most representative databases for regulatory elements including TFs, TFBSs, nucleotide distribution matrices of the TFBSs, and regulated genes. The data are expertly annotated and manually curated from peer-reviewed and experimentally proved publications. Other annotation databases of TF families and binding domains are also available (e.g. PROSITE [9], Pfam [10]). It is even more difficult and time consuming to extract high-resolution 3D TF-TFBS complex structures with X-ray crystallography or Nuclear Magnetic Resonance (NMR) spectroscopic analysis. Nevertheless, the high-quality TF-TFBS binding structures provide valuable insights into verifications of putative principles of binding. The Protein Data Bank (PDB) [11] serves as a representative repository of such experimentally extracted protein-DNA (in particular TF-TFBS) complexes with high resolution at atomic levels. However, the available 3D structures are far from complete. As a result, there is strong motivation to have automatic methods, particularly, computational approaches based on existing abundant data, to provide testable candidates of novel TF domains and/or TFBS motifs with high confidence to guide and accelerate the wet-lab experiments.

2.1.2 Existing Methods

The first attempt of computational methods related to TF-TFBS bindings was to discover the motifs of TF domains and TFBSs separately. Moreover, researchers have been trying hard to generalize the one-to-one binding codes from existing 3D structures. Data mining methods have also been proposed with feature transformations and machine learning to decipher complicated binding rules. They are briefly described as follows:

Motif discovery: TF domains and TFBSs sequences are somewhat conserved due to their functional similarity and importance. By exploiting conservation in the sequences, Bioinformatics methods called motif discovery save some of the expensive and laborious laboratory experiments. Motif discovery [6] can be categorized into two types: (a) motif matching and (b) *de novo* motif discovery. (a) Motif matching is to identify putative TF domains [9, 10] or TFBSs [41] based on motif knowledge obtained from annotated data. (b) *de novo* motif discovery predicts conserved patterns without knowledge on their appearances, based on certain motif models and scoring functions [42, 43] from a set of protein/DNA promoter sequences with similar regulatory functions. While *de novo* motif discovery is successful for well conserved TF functional domain motifs, the counterpart for TFBSs remains very challenging with poor performances on real benchmarks [6, 44, 45]. A significant limitation of motif discovery is the lack of linkage between the binding counterparts for revealing TF-TFBS relationships.

One-to-one binding codes: Numerous studies have been carried out to analyze existing protein-DNA binding 3D structures comprehensively [40, 46, 47] or with focus on specific families [39] (e.g. zinc fingers [48]). Various properties have been discovered concerning, e.g., bonding and force types, TF conservation and mutation [39], and bending of the DNA [46]. Some are already applicable to predict binding amino acids on the TF side, e.g. [49]. However, annotated data are far from complete. Alternatively, researchers have sought hard for general binding "codes" between proteins and DNA, in particular the one-to-one mapping between amino acids from TFs and nucleotides from TFBSs. Despite many proposed one-one binding propensity mappings [39, 50, 51], it has come to a consensus that there are no simple binding "codes" [38].

Data mining: In the hope of better understanding for protein-DNA bindings, many data mining approaches have also been proposed [33]. Researchers employ and transfer additional detailed information such as base compositions, structures, thermodynamic properties [34, 35] as well as expressions [36], into sophisticated features to fit into certain data mining techniques. Although some approaches may provide interpretable rules, most of them have stringent data requirements which cannot be obtained trivially. Existing data beyond sequences are also insufficient and limited for practitioners. These methods usually extract complicated features rather than working on interpretable data directly. Many data mining techniques, such as neural networks, support vector machines (SVM) [37], and regressions [33], may generate rules which are not trivial to interpret. Furthermore, many data mining approaches are based on specific families or particular datasets, where the generality of the results are limited. On the other hand, sequences serve as the most handy primary data which carry important information for protein-DNA bindings [38]. It is desirable to make use of the large-scale and comprehensive sequence data to mine explicit and interpretable TF-TFBS binding rules.

2.1.3 Related Databases

2.1.3.1 TRANSFAC - Experimentally Determined Database

Up to now, TRANSFAC is the state-of-the-art database on experimentally proven transcription regulatory elements [52]. It has been developed since 1996 [53]. BIOBASE GmbH in Germany is now in charge of it. It is a database on transcription factors, binding sites, and regulated genes for all species. Positional weight matrices are derived using the above data. Up to TRANSFAC 2008.3, it have 11,683 transcription factor data, 30,227 binding site data and 33,159 regulated gene data according to the BIOBASE company website (http://www.gene-regulation.com /faqs/TFsubscription.html). At the same time, TRANSFAC also have different modules to complement its main core data: PathoDB, S/MARt DB, TRANSPATH, CYTOMER, TRANSPro, and TRANSCompel [52, 54].

PathoDB is a database storing information about pathologically relevant mutations in transcription factor genes or in their binding sites. It comprises numerous cases of defective transcription factors or mutated transcription factor binding sites, which are known to cause pathological defects. S/MARt DB has been built as a relational database which comprises Scaffold matrix attached regions (S/MARs) and S/MAR-binding protein data. Scaffold matrix attached regions (S/MARs) have been shown to affect transcriptional regulation as a distinct class of cis-acting elements, but only recently they were shown to be targeted by several chromatin remodeling factors and thereby to influence gene expression. Besides S/MARs, a lot of transcription factors are regulated in response to extracellular signaling molecules. To capture such kind of information, TRANSPATH is built as a database on the signal transduction data. CYTOMER is a relational database that comprises tables for organs, cell types, physiological systems and developmental stages. TRANSPro is a database on eukaryotic promoter sequences. Last but not least, TRANSCompel is a database on composite regulatory elements affecting gene transcription in eukaryotes. Composite regulatory elements consist of two closely situated binding sites for distinct transcription factors, and provide cross-coupling of different signaling pathways.

The TRANSFAC data is expertly annotated, manually curated from peer-reviewed publications, which have been experimentally proved. The curators search on the literatures and input the digested data into database via input clients, using controlled vocabulary. A quality measure is assigned to each database entry. The quality measure is assigned based on the quality of the related literature. For example, the absence of errors and redundancy, the completeness, unambiguousness, high integration with other data sources, and existing commentary. TRANSFAC has been linked to a number of databases. For instance, TRANSFAC contains cross-references to the EMBL data library and to the

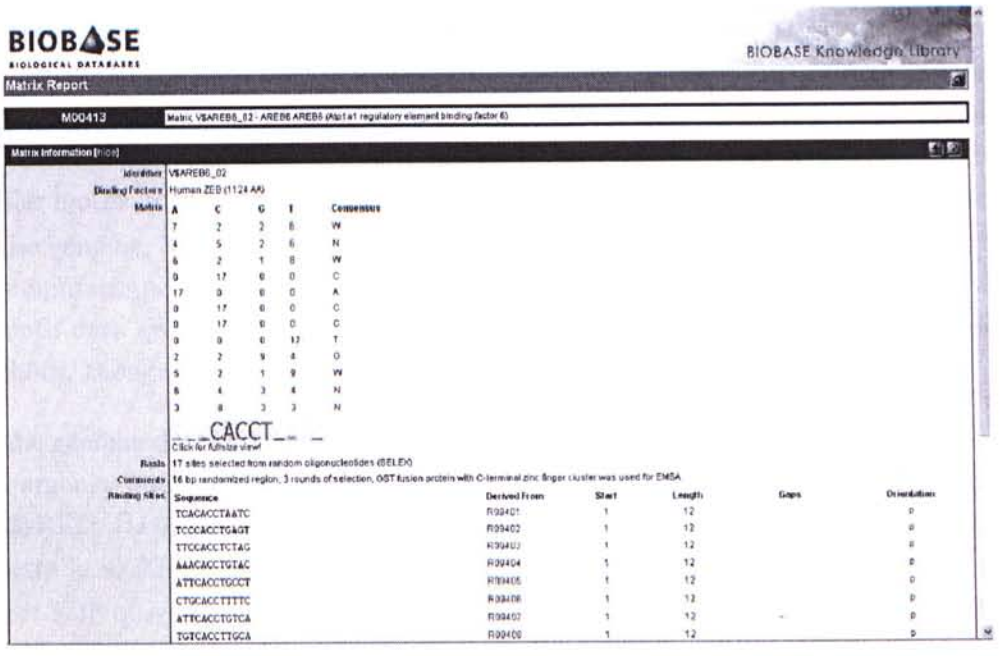


FIGURE 2.2: A screenshot of TRANSFAC

SwissProt database. To browse the database, web interfaces have been developed in TRANSFAC for users to query and browse the data.

On the application side, TRANSFAC has a tool called Match which is a web-based tool to identify transcription factor binding sites by weight matrix search in DNA sequences. Match is integrated in TRANSFAC Professional and equipped with positional weight matrices for analysis. Besides Match, Patch is also provided as a web-based tool which identifies transcription factor binding sites by pattern matching in DNA sequences. The transcription factor binding sites of TRANSFAC Professional are used as search patterns. Third-party tools have been integrated to enhance its usability. For instance, BLAST is connected and called 'TfBlast' in TRANSFAC. It is a search tool for sequence homology search in the TRANSFAC Transcription Factor data. Other miscellaneous tools have also been connected. Details can be referred to the TRANFAC website.

TRANSFAC public version is free for online use at http://www.gene-regulation.com/pub/databases.html, whereas TRANSFAC professional version needs fee subscriptions. The professional version has considerably large data amount comparing to the public version. Only flat files can be downloaded in the professional version.

2.1.3.2 cisRED - Computational Determined Database

cisRED was published in 2006 [55] and developed by Canada's Michael Smith Genome Sciences Centre in Canada. cisRED is a database on computationally predicted regulatory element data in eukaryotes. The real genome data is downloaded from the

Internet. Genome-scale prediction approaches are then applied to scan over the real genome data to obtain the computationally predicted binding site data (motif data). These approaches involve multiple motif discovery methods, optimal p-value tuning using known data (experimentally determined data) and clustering methods to identify similar motifs. cisRED contains 236k motif data for human genome, 223k motif data for mouse genome, 116k motif data for rat and 158k motif data for *C.elegans*. As the data is computationally derived, the motif data is not experimentally proved. The quality of motif data greatly depends on the accuracy and sensitivity of those motif discovery methods, though these methods are applied to the real genome data.

As the genome data is downloaded from others, the genome data is linked to the respective organizations such as Ensembl. On the other hand, as the motif data is predicted by cisRED, there is no external reference links to them. The motif data can be viewed directly in cisRED's user interfaces. UCSC genome browser can be used for browsing. Direct SQL query can also be sent to the cisRED database. Though not directly related to the application field of cisRED, some applications can be downloaded via the cisRED web interfaces for further analysis. For example, Sockeye and HitPlotter. cisRED is available at http://www.cisred.org/. All data can be downloaded as MySQL files.

2.1.3.3 ORegAnno - Community Driven Database

With the open-source and open-access atmosphere wide-spreads on the Internet in recent years, a database called ORegAnno appeared in 2008[56]. ORegAnoo is an open-access community-driven database and literature curation system for regulatory annotation. ORegAnno is a database on regulatory regions of all species. As of November 2008, ORegAnno contains 37469 regulatory region data, 14607 binding site data, 175 regulatory polymorphism data and 7 regulatory haplotype data. One distinguished feature of ORegAnno is its data acquisition methods. A queue called "Publication queue" has been implemented in ORegAnno. If some users feel that the publications give some results suitable for annotations, the users can freely input publications into this queue. On the other hand, users can also choose to check out publication from the queue and help ORegAnno perform manual curation and data extraction. Text-mining queue has also being developed to automate the publication insertion. Basic error checking is performed when inserting/updating data such as checking whether there is an existing data record with the same value. Existing data entry can be updated, commented, and scored. It is scored positive if users verify it as correct, it is scored negative if a problem is found. Nevertheless, in case of wrong annotation, the curated data entry can also be replaced or commented by other 'Validator' users. Such highly dynamic activities can ensure that the system maintain an acceptable quality, though may not be the highest quality. It is just something like Wikipedia vs Commercial encyclopedia (ORegAnno vs TRANSFAC). Besides, an ORegAnno evidence ontology has been created to capture

critical details from primary experiments so that each data entry can be described using a controlled set of vocabularies.

The records are well cross-referenced. Each data entry is cross-referenced to UCSC, Ensembl, EBI, PubMed, and dbSNP. ORegAnno can also been accessed through web services using Perl SOAP modules. Basic web query interfaces are available on the ORegAnno website. Ensembl and UCSC genome browser are also connected in ORegAnno. Sequence fetchers and scanners are equipped in ORegAnno. Several external applications such as Sockeye are connected in ORegAnno. ORegAnno is available at http://www.oreganno.org/. All data can be downloaded as XML files. MySQL dump can be downloaded by addressing to the authors.

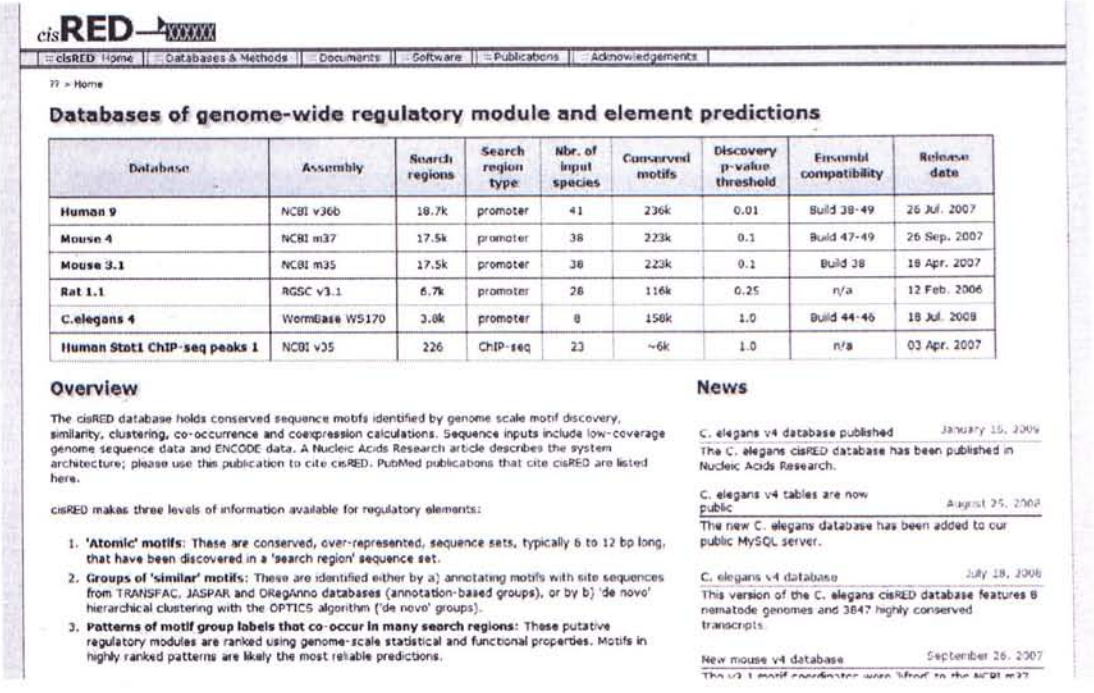


FIGURE 2.3: A screenshot of cisRED

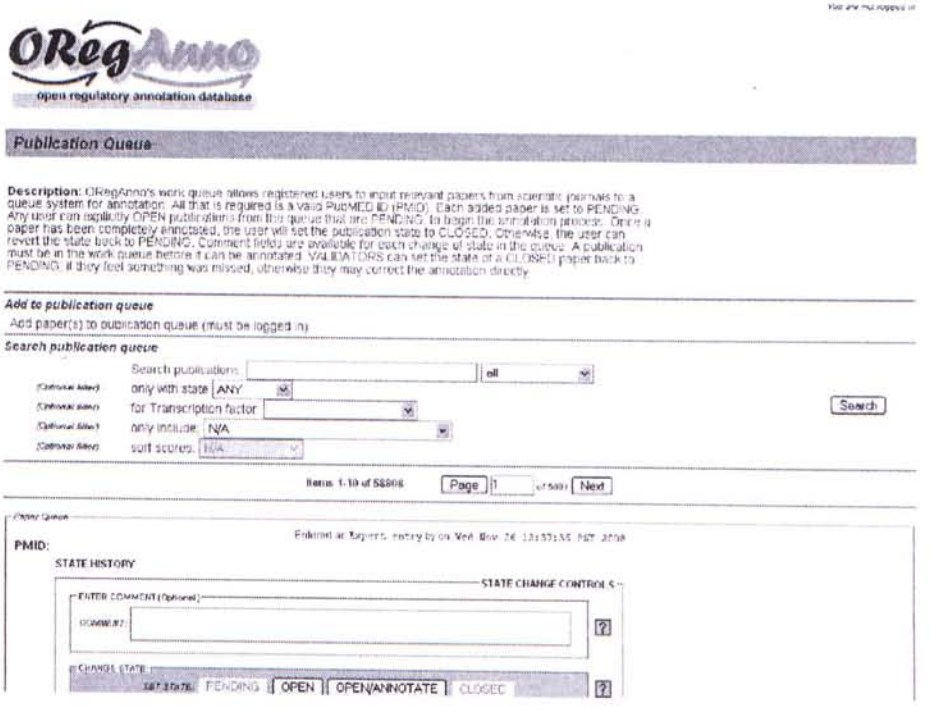


FIGURE 2.4: A screenshot of the publication queue in ORegAnno

2.2 Evolutionary Algorithms

Algorithm 1 A Typical Evolutionary Algorithm

Choose suitable representation methods;

$P(t)$: Parent Population at time t
$O(t)$: Offspring Population at time t

$t \leftarrow 0$;
Initialize $P(t)$;
while not termination condition **do**
  $temp$ = Parent Selection from $P(t)$;
  $O(t+1)$ = Crossover in $temp$;
  $O(t+1)$ = Mutate $O(t+1)$;
  if overlapping **then**
    $P(t+1)$ = Survival Selection from $O(t+1) \cup P(t)$;
  else
    $P(t+1)$ = Survival Selection from $O(t+1)$;
  end if
  $t \leftarrow t+1$;
end while

Good individuals can then be found in $P(t)$;

Evolutionary algorithms [1] draw inspiration from the nature. An evolutionary algorithm starts with a population randomly initialized. The population then evolve across several generations. In each generation, some individuals are selected to become parent individuals. They crossover with each other to form new individuals, called offspring individuals. Some of the offspring individuals may then undergo certain mutations. After that, the algorithm selects individuals according to the survival selection scheme designed. If the algorithm is overlapping [57], then both parent and offspring populations will participate in the survival selection. Otherwise, only the offspring population will participate in the survival selection. The selected individuals then survive to the next generation. Algorithm 1 briefly outlines a typical evolutionary algorithm.

In this section, we follow the unified approach proposed by De Jong [57]. The design of an evolutionary algorithm can be partitioned into several modules: representation, parent selection, crossover operators, mutation operators, survival selection, and termination condition :

- Representation involves genotype representation and genotype-phenotype mapping. [57]. For instance, we may represent an integer (phenotype) as a binary

[1] Indeed, evolutionary algorithms can also be classified as parallel point search methods which iteratively sample a predefined space to maximize an objective function. A point is an individual. A population refers to a group of points. An iteration is called a generation. The search space scored by the objective function is called the fitness landscape.

array (genotype) like '19' as '10011' and '106' as '1101010'. If we mutate the first bit, then we will get '3' (00011) and '42' (0101010). In the examples, even we have mutated one bit in the genotype, the phenotype may vary very much. Thus we can see that there are a lot of considerations in the mapping. Thus representation is one of the most important parts in design. It will be further discussed in section 2.2.1.

- Parent selection aims to select good parent individuals for crossover, where the goodness of a parent individual is typically proportional to its fitness. Thus most parent selection schemes prefer giving more opportunities to the fitter parent individuals and vice versa such that fitter offspring individuals are more likely to be generated. Details can be referred to section 2.2.2.

- Crossover operators simulate the reproduction mechanism in the nature. Thus they, with mutation, are sometimes called reproductive operators. In general, a crossover operator combines two individuals to form a new individual. It tries to split an individual into parts and then assemble the parts into a new individual. Details can be referred to section 2.2.3.

- Mutation operators simulate the mutation mechanism in which some parts of a genome undergoes random changes in the nature. Thus, as a typical modeling, a mutation operator in an evolutionary algorithm changes parts of the genome of an individual. In a typical run, mutations can be regarded as exploration mechanisms to balance the exploitation power of crossover operators. Details can be referred to section 2.2.4.

- Survival selection aims to select a subset of good individuals from a set of individuals, where the goodness of a individual is also proportional to its fitness in a typical case. Thus survival selection mechanism is somehow similar to parent selection mechanism. In a typical framework like 'EC4' [57], most parent selection mechanisms can be re-applied in survival selection. Details can be referred to section 2.2.5.

- Termination condition refers to the condition at which an evolutionary algorithm should end. Details can be referred to section 2.2.6.

To design an evolutionary algorithm, we need to choose a suitable combination of the above modules. However, several choices are available for each module. In total, there are hundreds of combinations available for a designer to choose. Thus some discussions are provided in section 2.2.7. Furthermore, some solid examples of evolutionary algorithms are described in section 2.2.8.

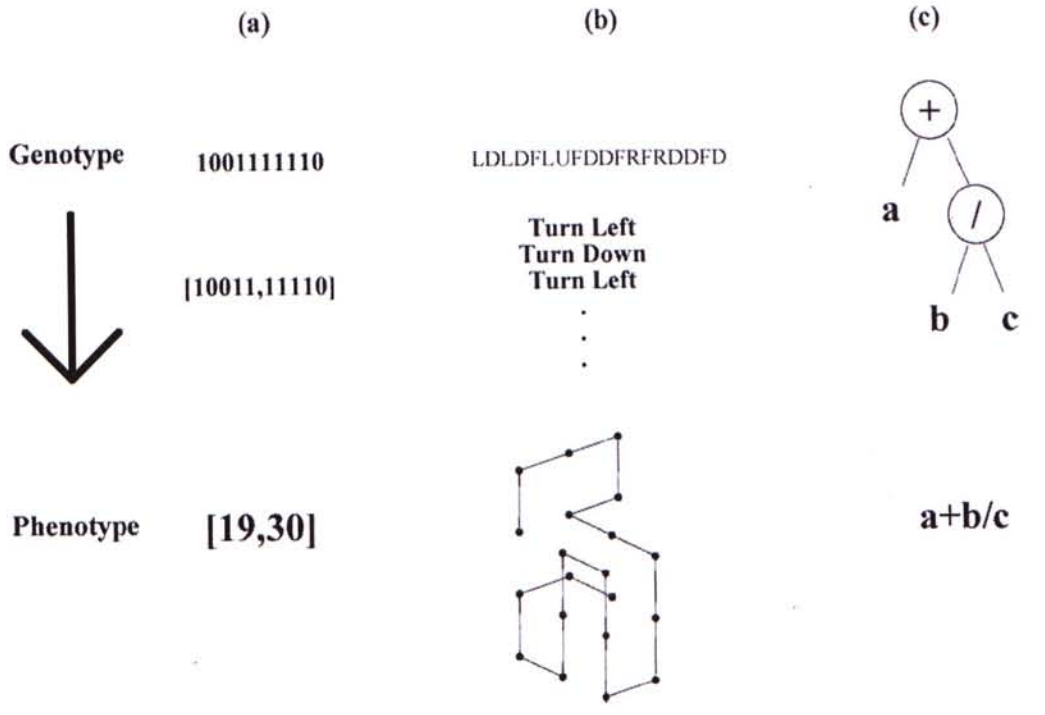


FIGURE 2.5: Representation Examples in Evolutionary Algorithms. (a) Integer representation (b) Protein structure representation on a lattice model (c) Tree representation for a mathematical expression

2.2.1 Representation

Representation involves genotype representation and genotype-phenotype mapping. In most cases, designers try to keep a genotype representation as compact as possible while keeping it as close to the corresponding phenotype representations as possible such that the metrics, say distance, in the genotype space can be mapped to those in phenotype space without losing much semantic information.

In general, there are many types of representations that an evolutionary algorithm can employ like fixed-length linear structures, variable-length linear structures, and tree structures ... To be concrete, Figure 2.5 depicts three examples. Figure 2.5a is one of the representations for a vector of integers. We can observe that its genotype is a binary array with length equal to 10. To map it into the phenotype space, the first 5 binary digits (10011) are mapped to the first element (19) of the vector whereas the remaining 5 binary digits (11110) are mapped to the second element (30) of the vector. Figure 2.5b is the relative encoding representation for a protein on a lattice model [58]. Its genotype is an array of moves and its length is set to the length of the amino acid sequence of the protein. The array of moves encodes the relative positions of amino acids from their predecessor amino acids. Thus we need to simulate the move sequence to get the 3D structure of the protein (phenotype) for further evaluations. Figure 2.5c is the tree representation for a mathematical expression. Obviously, such tree structure is a variable length structure, which has the flexibility in design. If the expression is short,

it can be shrunk during the evolution. If the expression is long, it can also be expanded during the evolution. Thus we can observe that the structure has an advantage over the previous representations. Nevertheless, there is no free lunch. It is also harder for people to implement and translate it into the corresponding phenotype. For example, the corresponding mathematical expression in infix notation format.

2.2.2 Parent Selection

Parent selection aims to select good parent individuals for crossover, where the goodness of a parent individual is typically proportional to its fitness. Thus most parent selection schemes prefer giving more opportunities to the fitter parent individuals and vice versa. Here are some methods:

- Fitness Proportional
  The scheme is sometimes called roulette wheel selection. In the scheme, the fitnesses of all individuals are summed. Once summed, the fitness of each individual is divided by the sum. The ratio then becomes the probability for each individual to be selected. If an algorithm needs to select an individual, then it will select the individuals based on the probability calculated.

- Rank Proportional
  Unlike fitness proportional scheme, the rank proportional scheme does not depend on the actual values of the fitnesses of the individuals. It is a double-edged sword. On the positive side, it can help us prevent the domination of very high fitness values. On the negative side, it imposes more computational costs to rank individuals such that the individual with a higher rank can be given a higher probability to be selected.

- Uniform Deterministic
  Every individual is selected.

- Uniform Stochastic
  Every individual is given equal probability to be selected.

- Binary Tournament
  Actually, there are many tournament selection schemes available. In this section, the most basic one, binary tournament, is described. In each binary tournament, two individuals are randomly selected and competed with each other by fitness. The winner is then selected. It is repeated until all vacancies are filled.

- Truncation
  Fitter individuals are selected deterministically when there is a vacancy for selection. In other words, worse individuals are never selected. For example, if there are

100 individuals and 50 slots are available for selection, the top 50 fittest individuals will be selected.

2.2.3 Crossover Operators

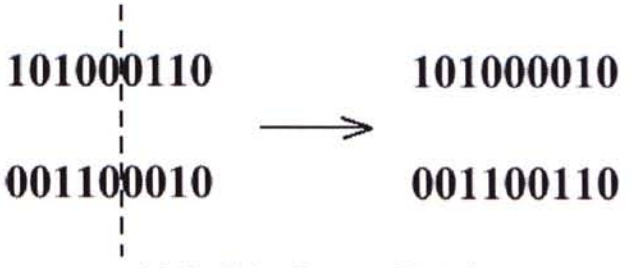
Crossover operators simulate the reproduction mechanism in the nature. Thus they, with mutation, are sometimes called reproductive operators. In general, a crossover operator combines two individuals to form a new individual. It tries to partition an individual into parts and then assemble the parts of two individuals into a new individual. The partitioning is not a trivial task. It depends on the representation the individuals have used. Thus it is not hard to imagine that crossover operators are representation-dependent. Nevertheless, without loss of generality, a list of crossover operators commonly used is shown below:

- One Point Crossover
  One point crossover is a commonly used crossover operator because of its simplicity. Given two individuals, it randomly chooses a cut point in their genomes. Then it swaps the parts after (or before) the cut point between the two genomes. Figure 2.6a depicts an example.

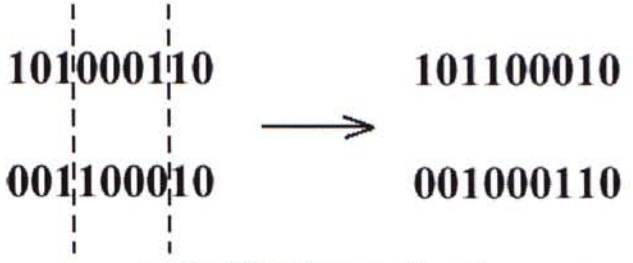
- Two Point Crossover
  Two point crossover is also a commonly used crossover operator because people argue that one point crossover has a positional bias toward the terminal positions. For instance, when making a one point crossover in Figure 2.6a, the rightmost (or leftmost) bit is always swapped because most cut points are before the bit. Thus the bit is mandated to be swapped, making it unable to stay after every one point crossover operation. Thus people propose two point crossover to avoid the positional bias. Figure 2.6b depicts an example.

- Uniform Crossover
  Uniform crossover is a general one. For each gene, it gives a uniform probability to be swapped.

- Blend Crossover
  Blend crossover is commonly used in real number optimization. Instead of swapping genes, it tries to blend two genes together and get the intermediate values. For instance, if we are going to make a crossover between two vectors [1 2 3] and [4 5 6], then the blended vector will be [2.5 3.5 4.5]. But, of course, the previous case is a typical example. Weights are sometimes given.



(A) One Point Crossover Example



(B) Two Point Crossover Example

FIGURE 2.6: One Point Crossover and Two Point Crossover Examples

2.2.4 Mutation Operators

Mutation operators simulate the mutation mechanism in which some parts of a genome undergoes random changes in the nature. Thus, as a typical modeling, a mutation operator in an evolutionary algorithm changes parts of the genome of an individual probabilistically. Again, similar to crossover operators, mutation operators are representation dependent. Nevertheless, without loss of generality, a list of commonly used mutation operators is shown:

- Bitflip Mutation
  Bitflip mutation is commonly used in binary genomes. Specified by a pre-defined probability, each bit in a binary genome is probabilistically inverted.
- Random Mutation
  Random mutation is generalized from bitflip mutation. It can be applied in many genomes. Specified by a pre-defined probability, each gene in a genome is probabilistically changed to a random value within the search space of the gene.
- Delta Mutation
  Delta mutation is commonly used in real number genomes. Specified by a pre-defined probability, each real number in a real number genome is probabilistically incremented/decremented by a certain step size (called delta), where the step size is set manually. But, of course, it is also straightforward to make the step size adaptive, like the trial vector generations in differential evolution [59].
- Gaussian Mutation
  Gaussian mutation is also commonly used in real number genomes. Like delta

mutation, each real number in a real number genome is probabilistically incremented/decremented by a step size. But the difference is that the step size is modeled as a Gaussian distribution. [57].

2.2.5 Survival Selection

Survival selection aims to select a subset of good individuals from a set of individuals, where the goodness of an individual is also proportional to its fitness in a typical case. Thus survival selection mechanism is somehow similar to parent selection mechanism. In a typical framework like EC4 [57], most parent selection mechanisms can be re-applied in survival selection like fitness proportional in section 2.2.2.

2.2.6 Termination Condition

Termination condition refers to the condition at which an evolutionary algorithm should end. Here is a list of the conditions commonly used:

- Number of Fitness Function Evaluations
  An evolutionary algorithm terminates when a certain number of fitness function evaluations has been reached.
- Number of Generations
  An evolutionary algorithm terminates when a certain number of generations has been reached.
- CPU Time
  An evolutionary algorithm terminates when it has been run for a certain amount of CPU times.
- Number of Births
  An evolutionary algorithm terminates when a certain number of individual births has been reached.
- Convergence
  An evolutionary algorithm terminates when further improvement on the population has not been observed.

2.2.7 Discussion

So far we have mentioned different modules and techniques in evolutionary algorithms. It is easy to be observed that we cannot simply adopt all of them in a single evolutionary algorithm. We must make a choice. Thus some discussions are provided in this section.

Parent selection aims to select good parent individuals for crossover, whereas survival selection aims to select a subset of good individuals from a set of individuals. Parent selection and survival selection are, indeed, not quite different. Their objectives are different, but their techniques are the same. Nevertheless, as described in section 2.2.2 and 2.2.5, many techniques exist. We must make a choice. Thus de jong [57] has done some experiments to help us rank the selection methods in terms of their greediness as follows (from the weakest to strongest):

- Uniform
- Fitness Proportional
- Rank Proportional and Binary Tournament
- Truncation

We can see that truncation is the greediest, whereas uniform is the weakest one. However it does not mean that we need to select truncation as the top choice because one of the beauties in evolutionary algorithms belongs to its parallel point search ability. If the greediest selection is applied without any diversity maintenance, evolutionary algorithms will easily suffer from genetic drift and lose parallel point search ability [60]. Thus we need to consider the balance between convergence and diversity needs to be well analysed before choosing a suitable selection scheme.

Regarding about the termination condition, many evolutionary algorithm researchers used different conditions. Nevertheless, as different algorithms perform different operations in one generation, it is unfair to set the termination condition as the number of generations. Number of births is also not a good choice because an evolutionary algorithm may exploit a single individual to perform many crossovers, mutations, and fitness evaluations. Alternatively, it is also unfair to adopt CPU time, since it substantially depends on the implementation techniques for different algorithms. For instance, the sorting techniques and the programming languages used. In contrast, a single fitness function evaluation may take several hours to complete [61]. The fitness function evaluation is always the performance bottleneck in practice. Thus the number of fitness function evaluations was set as the termination condition in the experiments to be described.

2.2.8 Examples

2.2.8.1 Genetic Algorithm

Genetic algorithm is the most classic evolutionary algorithm. It draws inspiration from the Darwin's Evolution Theory. The difference between genetic algorithm and evolutionary algorithm becomes blurred gradually nowadays. The words 'genetic algorithm' and

'evolutionary algorithm' are sometimes interchanged in use. Anyway, to clearly explain the working mechanism of a genetic algorithm, we use the canonical genetic algorithm [62] as a basis of genetic algorithm.

In the canonical genetic algorithm, each individual has a fixed-length binary array as its genotype. Then the fitness of each individual is divided by the average fitness to become the probability to be selected. Based on the calculated probability, the algorithm select parents for one point crossover to produce offspring individuals, which subsequently undergoes mutation. The offspring individuals become the population in the next generation and so and so forth. Using the previous terminology, the canonical genetic algorithm is a non-overlapping model since only the offspring individuals are selected for survival.

2.2.8.2 Genetic Programming

Genetic programming is indeed a special type of genetic algorithm. The difference lies in the representation method. Genetic programming uses trees as genotype to represent programs or expressions. (Figure 2.5c depicts an example). The typical selection schemes of evolutionary algorithms can still be used as parent selection and survival selection in genetic programming. What make genetic programming more special are their crossover and mutation operators. The crossover and mutation operators become some operators on tree structure. For instance, swapping sub-trees between two trees and random generation of sub-trees . A list of common crossover and mutation operators are tabulated in Table 2.1.

	Name	Description
Crossover	Subtree Exchange Crossover	Exchange subtrees between individuals
	Self Crossover	Exchange subtrees within an individual
	Module Crossover	Exchange modules between individuals
	SCPC	Exchange subtrees if coordinates match exactly
	WCPC	Exchange subtrees if coordinates match approximately
Mutation	Point Mutation	Change the value of a node
	Permutation	Change the argument order of a node
	Hoist	Use a subtree to become a new individual
	Expansion Mutation	Exchange a subtree against a terminal node
	Collapse Subtree Mutation	Exchange a terminal node against a subtree
	Subtree Mutation	Replace a subtree by another subtree
	Gene Duplication	Replace a subtree by a terminal

TABLE 2.1: A list of crossover and mutation operators [2]

2.2.8.3 Differential Evolution

Differential Evolution [59] was first proposed by Price and Storn in 1995 [63]. It demonstrated great potential for real function optimization in the subsequent contests [64, 65].

Without loss of generality, a typical strategy of differential evolution (DE/rand/1) [66] is shown in Algorithm 1.

Algorithm 2 Differential Evolution

```
P_t: Population at time t
TP: Transient population

t ← 0;
Initialize P_t;
Evaluate P_t;
while not termination condition do
    TP ← ∅;
    for ∀indiv_i ∈ P_t do
        Offspring ← TRIALVECTORGENERATION(indiv_i);
        Evaluate Offspring;
        if Offspring is fitter than indiv_i then
            Put Offspring into TP;
        else
            Put Parent into TP;
        end if
    end for
    t = t + 1;
    P_t ← TP;
end while
```

For each individual $indiv_i$ in a generation, the algorithm randomly selects three individuals to form a trial vector. One individual forms a base vector, whereas the value difference between the other two individuals forms a difference vector. The sum of these two vector forms a trial vector, which recombines with $indiv_i$ to form an offspring. Replacing the typical crossover and mutation operation by this trial vector generation, manual parameter tuning of crossover and mutation is no longer needed. It can provide differential evolution a self-organizing ability and high adaptability for choosing suitable step sizes which demonstrated its potential for continuous optimization in the past contests [64]. A self-organizing ability is granted for moving toward the optima. A high adaptability is achieved for optimizing different landscapes [66]. With such self-adaptability, differential evolution is considered as one of the most powerful evolutionary algorithms for real function optimization. It is still a hot research field in which scholars are exploring for its power in different fields. For example, mechanical engineering design [25] and nuclear reactor core design [31].

2.2.8.4 Evolution Strategy

Evolution Strategy was proposed in 1968 [67]. It was even older than genetic algorithm. Schwefel and Klockgether originally used evolution strategy as a heuristic to perform several experimental optimizations in air flow. They found that evolution strategy was

better than other discrete gradient-oriented strategy, which raised people's interests in evolution strategy.

Comparing to the previous evolutionary algorithms, evolution strategy draws less inspiration from the nature. Instead, it was artificially created as a numerical tool for optimization. Thus the structure of evolution strategy is quite different from the other evolutionary algorithms. For example, the mutation step size and probability. Evolution strategy calls them as endogenous parameters encoded in the genome of an individual. Thus, besides the gene values, a genome is also composed of the parameter settings which control the convergence progress of the whole algorithm. The notation evolution strategy uses is quite interesting. $(\mu/\rho^{+}_{,}\lambda) - ES$ denotes an evolution strategy with a specific parameter setting. μ denotes parent population size, ρ denotes the breeding size, $+$ denotes the algorithm is overlapping, , denotes the algorithm is not overlapping, λ denotes the offspring population size.

2.2.8.5 Swarm Intelligence

Ant Colony Optimization [68], Particle Swarm Optimization [69], and Bee Colony Optimization [70] ...etc are collectively known as Swarm Intelligence. Swarm intelligence is a special class of evolutionary algorithm. It does not involve any selection, birth, or death. Instead, it maintains a fixed-size population of individuals to search in a parameter space in each generation. After each generation, the individuals report their findings which are recorded and used to adjust the search strategy in the next generation. Thus, some of the algorithms were originally designed for shortest path finding. Nevertheless, people have further generalized them for other applications. For instance, Bi-Criterion Opitmization [71], Load Balancing in Telecommunication Network [72], Protein Folding Problem [73], and Power System [74].

2.3 Association Rule Mining

Association rule mining is one of the hot topics in data mining. It has become popular since 1990s. It not only helps in analyzing the relationship between different attributes/items, but also providing valuable information for people to make a right decision. This section briefly reviews the methods for association rule mining. Interested parties may find it useful in reading the general survey by Hipp et al. [75].

2.3.1 Objective

Association rule mining aims at finding association rules with minimal values in support and confidence from a database:

- **Database**: A set of transactions.
- **Transaction**: A binary vector with t[k]=1 if the item I_k is bought.
- **Association rule**: An expression of the form $X => I$ where X and I are two non-overlapping sets of items. Semantically, given X in some transactions, I is likely to be found in those transactions.
  - **Support**: The fraction of transactions which have the union of X and I.
  - **Confidence**: Among the transactions which have X, the fraction of those transactions which also have I.

2.3.2 Apriori Algorithm

Apriori algorithm proposed by Agrawal et al. [76, 77] is a classical approach to find out frequent itemsets (sets of items), outlined in Algorithm 2. It is a branch and bound algorithm for discovering association rules in a database. With its downward closure property, an optimal performance is guaranteed. In other words, given a pair of values of confidences and support, all the related association rules must be found by the algorithm. The algorithm first obtains frequent 1-itemsets. Iteratively, it uses the frequent n-itemsets (itemsets with n items) to generate all possible candidate (n+1)-itemsets. They are then evaluated for their supports [75]. If the support of an (n+1)-itemset is lower than a threshold, the (n+1)-itemset is removed. After the removal, the resultant (n+1)-itemsets are the frequent (n+1)-itemsets. The above procedure is repeated until an empty set is found.

Algorithm 3 Pseudocode of Apriori algorithm [76, 77]

data : A dataset of itemsets
L_n: Frequent n-itemsets
C_n: Candidate n-itemsets
x : An itemset
minsupport: Minimum Support
$i \leftarrow 1$;
Scan *data* to get L_i;
while $L_i \neq \emptyset$ **do**
  $C_{i+1} \leftarrow$ EXTEND(L_i);
  $L_{i+1} \leftarrow \emptyset$;
  for $x \in C_{i+1}$ **do**
    if $support(x) \geq minsupport$ **then**
      $L_{i+1} \leftarrow L_{i+1} \cap x$;
    end if
  end for
  $i \leftarrow i + 1$;
end while

Notes:
EXTEND(L_i) is the function "Candidate itemset generation procedure" stated in [76].
$Support(x)$ returns the support [75] of the itemset x.
A frequent n-itemset is the n-itemset which support is higher than *minsupport*.

2.3.3 Partition Algorithm

Savasere et al. has proposed Partition Algorithm [78], which logically divides the database into several non-overlapping partitions in main memory to generate a set of potentially large itemsets, which is a superset of the large itemsets, in the first scan. In the second scan, it calculates the actual supports of the potentially large itemsets to dig out the large itemsets. Thus only two scans on the database are needed.

2.3.4 DHP

In view of the expensive computation costs in candidate itemset generation, Park et al. [79] has proposed an algorithm DHP, which is especially effective in filtering unnecessary candidates by hashing techniques.

2.3.5 Sampling

Since most databases are quite large, it is trivial that sampling can come into play. Thus Toivonen [80] has proposed a sampling technique to mine association rules from a large database in a single pass. Though sampling, the technique can be remedied by a second pass to provide the exact performance.

2.3.6 Frequent Pattern Tree

To avoid repeated scanning on the database to check candidate items (or patterns), Frequent Pattern Tree (FP-tree) was proposed by Han et al. [81]. By two scans on a database, each transaction is mapped to one path in the FP-tree of the database. The transactions with the same items are mapped into a single path. Thus the frequent itemset information in each transaction is compactly and completely stored in the FP-tree. By keep tracking the nodes for each item in a header table, it can thus help us find out frequent itemsets, and thus association rules, from a database without scanning on the database repeatedly.

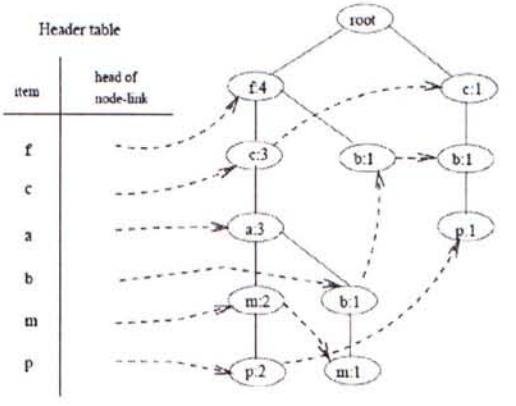


FIGURE 2.7: An example of FP-tree [82]

Chapter 3

Discovering Protein-DNA Binding Sequence Patterns Using Association Rule Mining

Protein-DNA bindings between transcription factors (TFs) and transcription factor binding sites (TFBSs) play an essential role in transcriptional regulation. Over the past decades, significant efforts have been made to study the principles for protein-DNA bindings. However, it is considered that there are no simple one-to-one rules between amino acids and nucleotides. Many methods impose complicated features beyond sequence patterns. Protein-DNA bindings are formed from associated amino acid and nucleotide sequence pairs which determine many functional characteristics. Therefore it is desirable to investigate associated sequence patterns between TFs and TFBSs.

With increasing computational power, availability of massive databases on DNA and protein data, and mature data mining techniques, we propose a framework to discover associated TF-TFBS binding sequence patterns in the most explicit and interpretable form from TRANSFAC. The framework is based on association rule mining with Apriori Algorithm. The patterns found are evaluated by quantitative measurements at several levels on TRANSFAC. With further independent verifications from literatures, Protein Data Bank (PDB), and homology modeling, there are strong evidences that the patterns discovered reveal real TF-TFBS bindings across different TFs and TFBSs, which can drive for further knowledge to better understand TF-TFBS bindings.

The chapter layout is as follows: the problem definition and proposed framework is presented in the section: **Materials and Methods**; experimental results and verifications are reported in the sections **Results and Analysis** and **Verifications** respectively; and finally we have the **Discussion** section for the approach. In addition, all the mathematical problem definitions are stated again in the appendix.

3.1 Materials and Methods

In this section, we propose a framework for mining, discovering, and verifying binding sequence patterns between TFs and TFBSs on TRANSFAC. The framework starts from data cleansing and transformation on TRANSFAC, and then applies association rule mining to discover TF-TFBS binding sequence patterns. Comprehensive 3D verifications and evaluations are carried out on PDB. Detailed bonding analysis is performed to provide strong supports to the discovered rules.

TF and TFBS data were downloaded and extracted from the flat files of TRANSFAC 2008.3 (a free public (older) version is also available [1]). The entries without sequence data were discarded. Since a TF can bind to one or more TFBSs, TFBS data were grouped by TF. TFBS sequences were extracted for each TF to form a TF dataset - a TF sequence and the corresponding TFBS sequences. Formally, m transcription factors $\{TF^1, TF^2, ..., TF^m\}$ are concerned. Each transcription factor TF^i chemically binds to several binding sites, which are collectively called transcription factor binding sites. Each of them is denoted by $TFBS^i_j$, the j-th transcription factor binding site which chemically binds to TF^i. For example, $TFBS^2_5$ denotes the 5-th transcription factor binding site which chemically binds to TF^2. Since different transcription factors can bind to different number of transcription factor binding sites, N^i denotes the number of the transcription factor binding sites which can chemically bind to TF^i. For example, we have the following transcription factor binding sites $\{TFBS^3_1, TFBS^3_2, ..., TFBS^3_{N^3}\}$ which can chemically bind to TF^3. They are formally defined as follows:

Definition 3.1 TF^i denotes the i-th transcription factor where $1 \leq i \leq m$.

Definition 3.2 $TFBS^i_j$ denotes the j-th transcription factor binding site which chemically binds to TF^i where $1 \leq i \leq m$ and $1 \leq j \leq N^i$.

To discover the binding sequence patterns between m transcription factors and their transcription factor binding sites. Sequence data is introduced. But, before doing so, some basic notations needs to be defined: $AAseq$ denotes a string with the amino acid alphabet, $DNAseq$ denotes a string with the DNA alphabet, and $kmer$ denotes a string with length $= k$. Thus $AAseq^i$ denotes the $AAseq$ of TF^i, whereas $DNAseq^i_j$ denotes the $DNAseq$ of $TFBS^i_j$.

The following sequence dataset is given for each transcription factor : the amino acid sequence of the transcription factor and the DNA sequences of the transcription factor binding sites which can chemically bind to the transcription factor. The dataset for TF^i is thus denoted by $TFdataset^i$: $\{AAseq^i, DNAseq^i_j | \forall j \in \mathbb{N}, j \leq N_i\}$. In total, $\{TFdataset^1, TFdataset^2, ..., TFdataset^m\}$ is given in this problem.

[1] http://www.gene-regulation.com/pub/databases.html

In the following subsections, Apriori algorithm for association rule mining is first introduced. We then elaborate how the algorithm is applied to protein-DNA binding pattern discovery. Finally we present how the data are preprocessed for the task with a running example.

3.1.1 Association Rule Mining and Apriori Algorithm

Association rule mining [76] aims at discovering frequently co-occurring items, called frequent itemsets, from a large number of data samples above a certain count threshold (minimum support) [75]. The support of an itemset is defined as the number of data samples where all the items in the itemset co-occur. In the case of protein-DNA binding, the binding domains of TFs can recognize and form strong bondings with certain sequence-specific patterns of the TFBSs. Therefore they are likely to co-occur frequently among the combinations between all possible TF and TFBS subsequences, and can be thus identified by association rule mining. In this study, we use the notation of k-mer (a subsequence with k amino acid or nucleotide residues) to represent a candidate item. A frequent TF-TFBS itemset is a TF k-mer and TFBS k-mer (the two k's can be different) pair, or simply a pair, co-occurring with a frequency no less than the minimum support in the TF-TFBS sequence records (TRANSFAC database).

Apriori algorithm proposed by Agrawal et al. [76] is a classical approach to find out frequent itemsets. It is a branch and bound algorithm for discovering association rules in a database. With its downward closure property, an optimal performance is guaranteed. The algorithm first obtains frequent 1-itemsets. Iteratively, it uses the frequent n-itemsets (itemsets with n items) to generate all possible candidate (n+1)-itemsets. They are then evaluated for their supports [75]. If the support of an (n+1)-itemset is lower than a threshold, the (n+1)-itemset is removed. After the removal, the resultant (n+1)-itemsets are the frequent (n+1)-itemsets. The above procedure is repeated until an empty set is found.

3.1.2 Discovering associated TF-TFBS sequence patterns

To formulate the TF-TFBS sequence pattern discovery problem into association rule mining, we have to transform the protein-DNA binding records into the formats of itemsets (k-mers). An illustrative example for the TF-TFBS binding records from TRANSFAC 2008.3 is shown in Fig.3.1. The TF (e.g. T01333 RXR-gamma) can bind to several TFBS DNA sequences. The DNA sequences may be different in lengths due to experimental methods and noises. Both the TF and TFBS sequences are chopped into overlapping short k-mers, as illustrated in Fig.3.2 (first part). They together with the corresponding reverse complements (e.g. GACCT and reverse complement: AGGTC) form one data sample. To generate the itemsets, all the k-mers are recorded in a binary

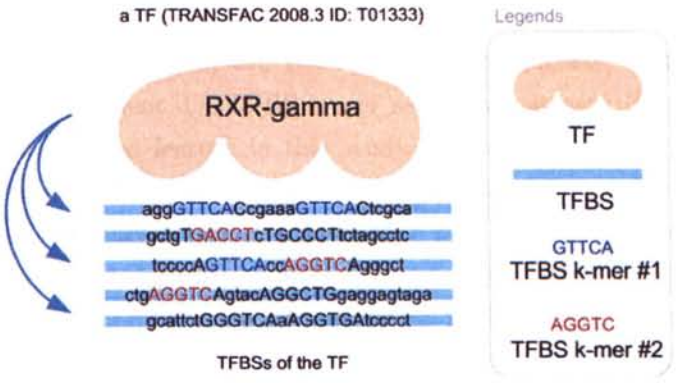


FIGURE 3.1: TFBS sequences of a TF (TRANSFAC 2008.3 ID: T01333)

array where appearing k-mers are marked 1; and 0 otherwise. Formally, a function F (Found) is defined:

Definition 3.3 If the kmer A is a substring of the string S, $F(A,S) = 1$. Otherwise, $F(A,S) = 0$.

Thus, the length of the array depends on the number of all possible TF k-mers and TFBS k-mers (Fig.3.2, second part). Since k is usually short (4-6), all the possible 4^k combinations of TFBS DNA k-mers can be adopted. However, it is computationally infeasible to obtain all the possible 20^k combinations of TF k-mers. Thus a data-driven approach is employed by scanning the whole TRANSFAC to obtain frequent TF amino acid k-mers.

Since there are multiple TFBSs for each TF (e.g. Fig.3.1), a question arises : How to define the "commonly found" TFBS k-mers of a TF. Without loss of generality, the majority rule [83] is applied. If the majority of a TF's TFBS sequences contains a certain DNA residue k-mer, then the k-mer is considered "commonly found". We set the majority to be 50% for TFBS k-mers. We only count the number of TFBS sequences in which a certain k-mer appears, in order not to be biased by multiple occurrences of the k-mer appearing in only a few TFBS sequences. Fig.3.1 illustrates an example where there are 5 TFBS sequences. The TFBS DNA k-mer AGGTC (or its reverse complement: GACCT) can be found in three of the TFBS sequences. The k-mer appears in 60% ($\frac{3}{5}$) of the TFBS sequences of the TF, and thus is considered "commonly found". On the other hand, GTTCA is not considered "commonly found" because it only appears in 2 out of the 5 (40%) TFBS sequences of the TF. Formally, a function CF^i (Commonly Found) is defined:

Definition 3.4 Let $Threshold_{CF} \in [0,1]$, and $DNAseq^i_j$ be the $DNAseq$ of $TFBS^i_j$. If $\frac{\sum_{j=1}^{N^i} F(A,DNAseq^i_j)}{N^i} \geq Threshold_{CF}$, $CF^i(A) = 1$. Otherwise, $CF^i(A) = 0$.

After all valid TF data samples are transformed into itemsets, Apriori algorithm is applied to generate frequent TF-TFBS k-mer sequence patterns (the links in Fig.3.2, second part). The special feature in this study is that the co-occurring pairs should contain both TF and TFBS k-mer items, as illustrated in the third part of Fig.3.2. In the current study, we only consider 1 TF k-mer with 1 TFBS k-mer in the frequent itemsets, but it is straightforward to generalize it to be multiple TF and TFBS k-mers in principle. The huge computational intensity for the generalization, when applied on the large TRANSFAC database, prevents us from doing so at this time. Finally the association rules are computed based on the confidence measurements for the frequent itemsets, which are defined as follows:

$$conf(kmer_{DNA} \Rightarrow kmer_{AA}) = \frac{support(kmer_{DNA} \cap kmer_{AA})}{support(kmer_{DNA})}$$

$$conf(kmer_{DNA} \Leftarrow kmer_{AA}) = \frac{support(kmer_{DNA} \cap kmer_{AA})}{support(kmer_{AA})}$$

where $conf(kmer_{DNA} \Rightarrow kmer_{AA})$ is called forward confidence, $conf(kmer_{DNA} \Leftarrow kmer_{AA})$ is called backward confidence, and $support(X)$ is the support of itemset X. For each association rule, its forward confidence measures the posterior probability that the corresponding amino acid k-mer can be found in a TF's sequence if the DNA k-mer is commonly found in the TF's TFBS sequences. Its backward confidence measures the posterior probability that the corresponding DNA k-mer can be commonly found in a TF's TFBS sequences if the amino acid k-mer is found in the TF's sequence. The minimum of them is taken as confidence. The higher the confidence, the better the association rule is (Fig.3.2, fourth part). The whole proposed approach is summarized in Fig.3.2.

3.1.3 Data Preparation

To apply the methodology on TRANSFAC, TF and TFBS data were downloaded and extracted from the flat files of TRANSFAC 2008.3 (a free public (older) version is also available [2]). The entries without sequence data were discarded. Since a TF can bind to one or more TFBSs, TFBS data were grouped by TF. TFBS sequences were extracted for each TF to form a TF dataset - a TF sequence and the corresponding TFBS sequences, and finally to be transformed into itemsets. To avoid sampling error, TF datasets with less than five TFBS sequences were discarded. Furthermore, the redundancy of TF sequences was removed by BLASTClust using 90% TF sequence identity [17]. Only one TF dataset was selected for each cluster. Note that we only used sequence data in TRANSFAC. None of the prior information (e.g. the binding domains of TFs) other than sequences was used. Importantly, it turns out that the results of the proposed

[2] http://www.gene-regulation.com/pub/databases.html

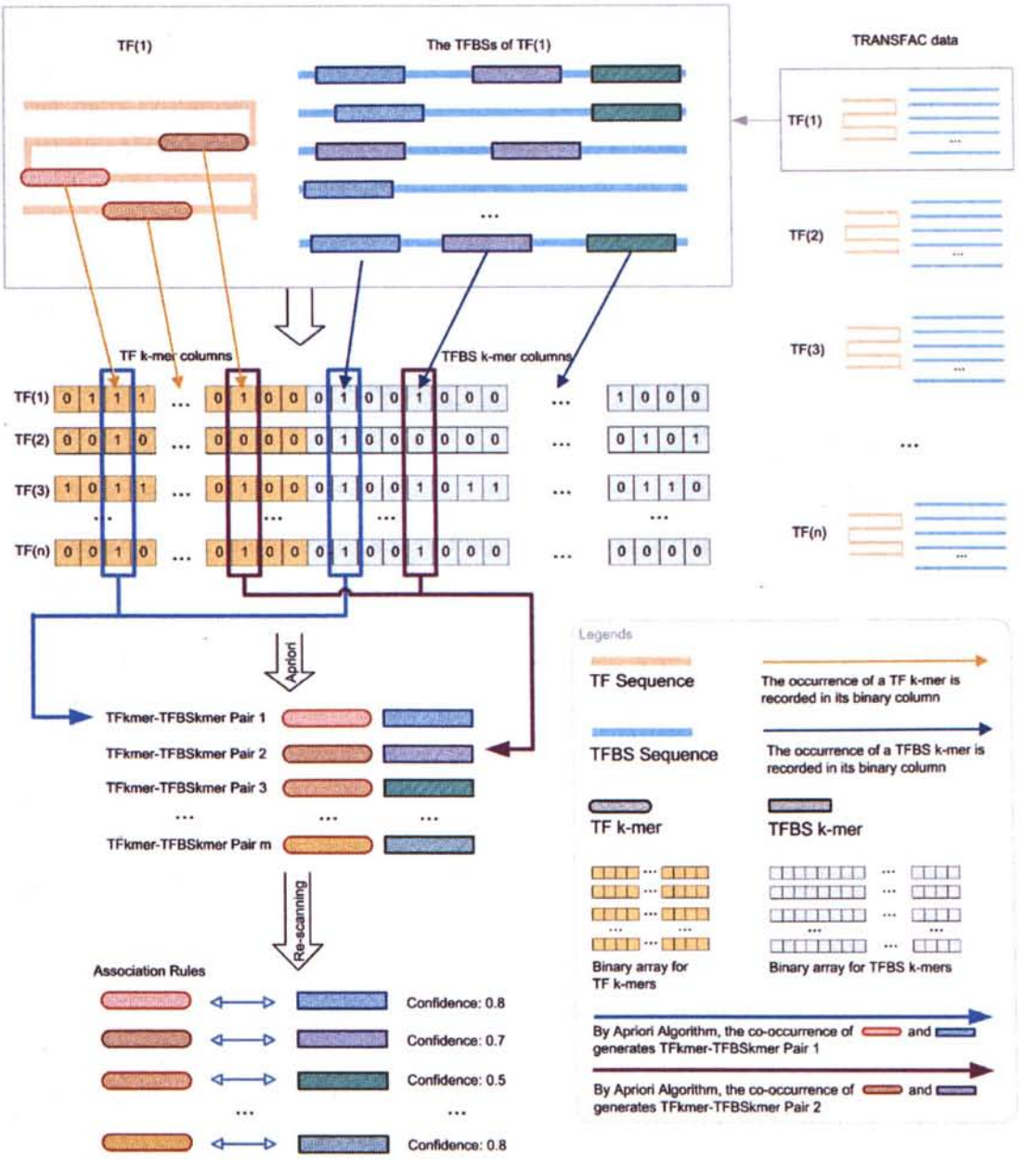


FIGURE 3.2: Flowchart of the proposed framework to discover association rules from TRANSFAC

approach can be verified by annotations, 3D structures from PDB, and even homology modeling as described in the subsequent sections.

After data preparation, the 631 TF datasets (listed in Table A.25 in the Appendix) were selected. The minimum support [75] was set to 7 TF datasets to avoid sampling error. For the values of k, we try 4-6 for both TF k-mers and TFBS k-mers, resulting in 9 (3 by 3) different combinations. In particular 256 DNA 4mers, 1024 DNA 5mers, and 4096 DNA 6mers were adopted for TFBS, whereas 99621 amino acid 4mers, 82561 amino acid 5mers, and 39320 amino acid 6mers were adopted for TF, as the frequent 1-itemsets.

Apriori algorithm was then applied to discover frequently co-occurring TF-TFBS k-mer pairs (2-itemsets). Finally, the resultant pairs were re-scanned in TRANSFAC to measure their forward and backward confidences [84]. Formally speaking, we are interested in the pairs of kmers in which one is found in the amino acid sequences of some transcription factors while the other is commonly found in the DNA sequences of their transcription factor binding sites. Such pairs are then called 'frequently co-occurring' defined as follows:

Definition 3.5 Let $Threshold_{Support} \in \mathbb{N}^+$, $k_1 \in \mathbb{N}^+$, and $k_2 \in \mathbb{N}^+$. A pair of kmers (A-B) is frequently co-occurring if $\frac{\sum_{i=1}^{m} CF^i(A) \times F(B, AAseq^i)}{m} \geq Threshold_{Support}$ where A is a $k_1 mer$ and B is a $k_2 mer$.

We assume that the pairs of kmers frequently co-occurring are the binding sequence patterns which we aims to find. Thus the problem definition is summarized as follows: Given $Threshold_{CF} \in [0, 1]$, $Threshold_{Support} \in \mathbb{N}^+$, $k_1 \in \mathbb{N}^+$, $k_2 \in \mathbb{N}^+$, we would like to find all pairs (A-B) frequently co-occurring where A is a $k_1 mer$ and B is a $k_2 mer$.

3.2 Results and Analysis

In this section, the discovered rules are reported, followed by analysis with different measurements.

3.2.1 Rules Discovered

Varying k from 4 to 6 for both TF k-mers and TFBS k-mers, we have obtained 9 sets of associated pairs. For each set of pairs, the forward and backward confidences of each pair were calculated. Then, the pairs in the same set were sorted by the minima of their forward and backward confidences in descending order. The 9 sets of rules (pairs) exhibit a similar trend that the number of rules decreases as the association criterion becomes more stringent (with higher confidence levels). The TFBS5mers settings in general show the most available rules when the confidence level is high (≥ 0.5), indicating more conserved and significant results. Therefore we focus on them and use TFBS5mer-TF5mer as the representative example throughout the paper. The results for all other settings are available in the Supplementary Data.

Without confidence pruning (confidence level = 0.0), the longer associated pattern pairs are the supersets of the shorter pairs according to the downward closure property. For instance, if the pair AAACA-HNLSL is found in the set of TFBS5mer-TF5mer pairs, then the two sub-pairs AAACA-HNLS and AAACA-NLSL must also be found in the set of TFBS5mer-TF4mer pairs. Note that such property is only applied when there is no confidence pruning. In other words, it is only applied to the second row of Supplementary Table 1 where confidence level = 0.0.

With confidence pruning (confidence level > 0.0), it can be observed that the degree of overlapping between different sets of pairs is decreased when the confidence level is increased. For instance, at the confidence level=0.1, 86.3% of the TFBS5mer-TF5mer pairs have all of their sub-pairs in the set of TFBS4mer-TF4mer pairs. At the confidence level=0.3, only 11.1% of the TFBS5mer-TF5mer pairs have all of their sub-pairs in the set of TFBS4mer-TF4mer pairs. Besides, it is interesting that the degree of overlapping is significantly decreased when TFBS k-mers are elongated. For example, at the confidence level=0.1, 92.6% of the TFBS4mer-TF5mer pairs (detailed in Supplementary Table 1) have all of their sub-pairs in the set of TFBS4mer-TF4mer pairs, whereas only 9.09% of the TFBS5mer-TF4mer pairs (Supplementary Table 1) have all of their sub-pairs in the set of TFBS4mer-TF4mer pairs.

The number of rules (pairs) discovered is summarized in Table 3.1. For instance, there are 70 TF5mer-TFBS5mer pairs without any further removal (in the N column) with both forward and backward confidences greater than or equal to 0.5. Considering direct

TABLE 3.1: Number of the TFBS5mer-TF5mer pairs across different confidence levels. (N: Number Of Pairs, N': Number of Pairs (duplicated pairs removed), N_m: Number of Merged Pairs, S: Mean & SD of the support of the pairs in N')

Confidence	N	N'	N_m	S
0.0	262	131	29	9.88±3.68
0.1	262	131	29	9.88±3.68
0.2	240	120	24	10.14±3.73
0.3	180	90	23	10.63±4.11
0.4	126	63	21	11.40±4.59
0.5	70	35	11	13.63±5.05
0.6	24	12	8	15.08±5.28
0.7	6	3	2	10.33±2.36
0.8	0	0	0	N/A
0.9	0	0	0	N/A
1.0	0	0	0	N/A

TABLE 3.2: Quantitative Measurements for the TFBS5mer-TF5mer pairs across different confidence levels. (ϕ: Mean & SD of ϕ-coefficient, L: Mean & SD of Lift, FC: Mean & SD of Forward Conviction, BC: Mean & SD of Backward Conviction)

Confidence	ϕ	L	FC	BC
0.0	0.49±0.11	17.92±7.34	1.89±0.67	3.50±2.29
0.1	0.49±0.11	17.92±7.34	1.89±0.67	3.50±2.29
0.2	0.51±0.11	18.32±7.46	1.94±0.68	3.51±2.30
0.3	0.54±0.10	19.81±7.79	2.02±0.64	3.46±2.31
0.4	0.58±0.09	21.41±8.53	2.23±0.66	3.61±2.40
0.5	0.64±0.07	22.57±10.46	2.49±0.70	4.35±2.65
0.6	0.71±0.06	25.80±13.76	3.33±0.57	4.21±2.55
0.7	0.79±0.03	42.07±14.87	3.70±0.29	4.87±0.00
0.8	N/A	N/A	N/A	N/A
0.9	N/A	N/A	N/A	N/A
1.0	N/A	N/A	N/A	N/A

and reverse complement TFBS DNA k-mers as equivalent, we further removed the duplicated pairs (e.g. leaving AGGTC-CEGCK and removing GACCT-CEGCK because AGGTC and GACCT are reverse complements). The results are shown in the N' column in Table 3.1. For instance, the 70 TF5mer-TFBS5mer pairs were reduced to 35 at confidence level = 0.5. Furthermore we found that most pairs could be merged together to form a longer pair. For instance, GGTCA-SGYHY and GGTCA-GYHYG could be merged to form a pair GGTCA-SGYHYG. Thus the pairs have been merged and the rule numbers are shown in the Nm column in Table 3.1. For instance, 35 TF5mer-TFBS5mer pairs are merged to form 11 merged pairs when the confidence level = 0.5.

3.2.2 Quantitative Analysis

To evaluate the number of TF datasets supporting each pair (support), the support for each pair was counted. In general, more supports are found when the confidence level is increased. For instance, the average support of the TFBS5mer-TF5mer pairs is generally increased when the confidence level is increased in the S column of Table 3.1. The overall results are summarized in Supplementary Table 4.

Support is considered the degree of co-occurring between a TF amino acid k-mer and a TFBS DNA k-mer. Forward and backward confidences consider the cases when either one of them is absent. Some may have questions about the remaining case. How about the case when both of them are absent? To take the case into account, ϕ-coefficients [85] was measured for each pair, as shown in the ϕ column in Table 3.2. The overall results are summarized in Supplementary Table 5. Most values are larger than 0.4, indicating that positive correlations exist among pairs.

Consider the following scenario: If a TFBS DNA k-mer and a TF amino acid k-mer are frequently co-occurring in the datasets, it will be very likely that they co-occur frequently merely by chance. To tackle such scenario, forward and backward confidences do play their important roles in pruning them. But for clarity, lift [86] which estimates the ratio of the actual support to the expected support was measured for each pair, where the expected support was calculated from the random model that the TFBS DNA k-mer is independent of the TF amino acid k-mer for each pair. For instance, the average lift for the TFBS5mer-TF5mer pairs are shown in the L column in Table 3.2. The overall results are summarized in Supplementary Table 6. Most values of the lift are larger than 5. Thus the DNA residue k-mer and the amino acid residue k-mer of most pairs co-occur at least 5 times more frequently than the prediction based on the independent assumption made by the lift measurement.

To estimate the validity of the pairs, both forward and backward convictions (the same directions as the forward and backward confidences respectively) [86] were measured for each pair. The measurements were averaged for each set of pairs. For instance, the average forward and backward convictions for the TFBS5mer-TF5mer pairs are shown in the FC and BC columns in Table 3.2. The overall results are summarized in Supplementary Tables 7 and 8. Most values are larger than one. The pairs commit fewer errors than the prediction based on the statistically independent assumption made by the measurements: forward and backward convictions. In other words, the pairs would have committed more errors if the association between its TFBS k-mer and TF k-mer had happened purely by chance.

TABLE 3.3: The set of TFBS 5mer-TF 5mer pairs (duplicated pairs removed and sorted in alphabetical order)

Confidence	Forward Confidence	Backward Confidence	Pairs	Confidence	Forward Confidence	Backward Confidence	Pairs	Confidence	Forward Confidence	Backward Confidence	Pairs
0.7	0.7	0.8	AAACA-HNLSL	0.4	0.4	0.7	AGGTC-CQYCR	0.3	0.5	0.3	GCCAC-ARRSR
0.5	0.5	0.7	AAACA-IRHNL	0.2	0.2	0.6	AGGTC-CVVCG	0.4	0.5	0.4	GCCAC-ESARR
0.5	0.5	0.6	AAACA-KPPYS	0.6	0.6	0.7	AGGTC-EGCKG	0.4	0.4	0.6	GCCAC-KQSNR
0.4	0.4	0.7	AAACA-NLSLN	0.2	0.2	0.7	AGGTC-FFRRT	0.4	0.6	0.4	GCCAC-NRESA
0.6	0.6	0.6	AAACA-NSIRH	0.2	0.2	0.8	AGGTC-FRRTI	0.4	0.4	0.6	GCCAC-QSNRE
0.5	0.5	0.6	AAACA-PPYSY	0.6	0.6	0.6	AGGTC-GCKGF	0.4	0.5	0.4	GCCAC-RESAR
0.4	0.4	0.6	AAACA-PYSYI	0.3	0.3	0.5	AGGTC-GFFKR	0.4	0.4	0.6	GCCAC-RKQSN
0.4	0.4	0.6	AAACA-QNSIR	0.4	0.4	0.5	AGGTC-GFFRR	0.4	0.4	0.5	GCCAC-RLRKQ
0.7	0.7	0.8	AAACA-RHNLS	0.3	0.3	0.6	AGGTC-KGFFK	0.4	0.4	0.5	GCCAC-RRSRL
0.5	0.5	0.8	AAACA-SIRHN	0.4	0.4	0.5	AGGTC-KGFFR	0.4	0.4	0.6	GCCAC-RSRLR
0.4	0.4	0.6	AAACA-WQNSI	0.4	0.4	0.9	AGGTC-RNRCQ	0.4	0.5	0.4	GCCAC-SARRS
0.4	0.4	0.6	AACAA-HNLSL	0.3	0.3	0.5	AGGTC-TCEGC	0.4	0.5	0.4	GCCAC-SNRES
0.3	0.3	0.6	AACAA-IRHNL	0.4	0.4	0.5	AGGTC-VCGDK	0.4	0.5	0.4	GCCAC-SRLRK
0.3	0.3	0.5	AACAA-NSIRH	0.2	0.2	0.5	AGGTC-VVCGD	0.6	0.6	0.8	GGTCA-CEGCK
0.3	0.3	0.7	AACAA-PMNAF	0.2	0.7	0.2	ATTAA-FQNRR	0.2	0.2	0.9	GGTCA-CGDKA
0.4	0.4	0.6	AACAA-RHNLS	0.2	0.6	0.2	ATTAA-IWFQN	0.5	0.5	0.6	GGTCA-CKGFF
0.3	0.3	0.7	AACAA-RPMNA	0.2	0.6	0.2	ATTAA-KIWFQ	0.3	0.3	0.9	GGTCA-CQYCR
0.3	0.3	0.7	AACAA-SIRHN	0.3	0.5	0.3	ATTAA-NRRMK	0.2	0.2	0.8	GGTCA-CVVCG
0.2	0.6	0.2	AAGGT-CKGFF	0.3	0.5	0.3	ATTAA-QNRRM	0.1	0.1	1	GGTCA-DLVLD
0.2	0.5	0.2	AATTA-FQNRR	0.2	0.7	0.2	ATTAA-WFQNR	0.5	0.5	0.8	GGTCA-EGCKG
0.3	0.3	0.3	AATTA-NRRAK	0.2	0.5	0.2	CACCC-GEKPY	0.2	0.2	0.8	GGTCA-FFKRS
0.4	0.4	0.5	AATTA-QNRRA	0.1	0.5	0.1	CACCC-HTGEK	0.2	0.2	0.8	GGTCA-FFRRT
0.3	0.3	0.7	AATTA-QVWFQ	0.1	0.5	0.1	CACCC-TGEKP	0.2	0.2	1	GGTCA-FRRTI
0.5	0.5	0.5	AATTA-VWFQN	0.5	0.5	0.5	CCACG-ARRSR	0.5	0.5	0.7	GGTCA-GCKGF
0.2	0.5	0.2	AATTA-WFQNR	0.5	0.5	0.6	CCACG-ESARR	0.2	0.2	0.5	GGTCA-GFFKR
0.5	0.5	0.7	ACGTG-ARRSR	0.3	0.3	0.7	CCACG-KQSNR	0.3	0.3	0.6	GGTCA-GFFRR
0.1	0.1	0.7	ACGTG-ERELK	0.2	0.2	0.6	CCACG-LRKQA	0.1	0.1	0.6	GGTCA-GYHYG
0.5	0.5	0.9	ACGTG-ESARR	0.6	0.6	0.6	CCACG-NRESA	0.1	0.1	1	GGTCA-ITCEG
0.2	0.2	0.8	ACGTG-KQSNR	0.3	0.3	0.6	CCACG-QSNRE	0.2	0.2	0.6	GGTCA-KGFFK
0.2	0.2	0.7	ACGTG-LRKQA	0.5	0.5	0.6	CCACG-RESAR	0.3	0.3	0.6	GGTCA-KGFFR
0.6	0.6	0.9	ACGTG-NRESA	0.2	0.2	0.7	CCACG-RKQAE	0.1	0.1	1	GGTCA-NRCQY
0.2	0.2	0.7	ACGTG-QSNRE	0.3	0.3	0.7	CCACG-RKQSN	0.1	0.1	1	GGTCA-RCQYC
0.5	0.5	0.9	ACGTG-RESAR	0.3	0.3	0.5	CCACG-RLRKQ	0.1	0.1	0.8	GGTCA-RNQCQ
0.1	0.1	0.7	ACGTG-RKQAE	0.3	0.3	0.6	CCACG-RRSRL	0.3	0.3	1	GGTCA-RNRCQ
0.2	0.2	0.8	ACGTG-RKQSN	0.3	0.3	0.6	CCACG-RSRLR	0.2	0.2	1	GGTCA-SCEGC
0.2	0.2	0.6	ACGTG-RLRKQ	0.5	0.5	0.6	CCACG-SARRS	0.1	0.1	0.5	GGTCA-SGYHY
0.2	0.2	0.7	ACGTG-RRSRL	0.5	0.5	0.6	CCACG-SNRES	0.3	0.3	0.6	GGTCA-TCEGC
0.2	0.2	0.8	ACGTG-RSRLR	0.4	0.4	0.4	CCACG-SRLRK	0.3	0.3	0.6	GGTCA-VCGDK
0.5	0.5	0.9	ACGTG-SARRS	0.5	0.5	0.5	CGGAA-LRYYY	0.2	0.2	0.7	GGTCA-VVCGD
0.5	0.5	0.9	ACGTG-SNRES	0.5	0.5	0.8	CTTCC-LRYYY	0.5	0.5	0.7	GTCAA-KYGQK
0.3	0.3	0.5	ACGTG-SRLRK	0.4	0.4	0.7	CTTCC-LWQFL	0.5	0.5	0.7	GTCAA-RKYGQ
0.6	0.7	0.6	AGGTC-CEGCK	0.4	0.7	0.4	GATAA-CNACG	0.5	0.5	0.7	GTCAA-WRKYG
0.3	0.3	0.8	AGGTC-CGDKA	0.4	0.7	0.4	GATAA-LCNAC	0.7	0.7	1	TGACA-NWFIN
0.6	0.7	0.6	AGGTC-CKGFF	0.6	0.7	0.6	GATAA-NACGL				

3.2.3 Annotation Analysis

If the pairs in our results are the actual binding cores between TFs and TFBSs, most of their TF amino acid k-mers should be inside DNA binding domains. Thus the TF amino acid k-mers were scanned in TRANSFAC to check whether they were within the annotated DNA binding domains. As stated in the previous section, the set of TFBS4mer-TF4mer pairs constituting all the pairs in the other sets by the downward closure property. Thus only the TF amino acid 4-mers of the set of TFBS4mer-TF4mer pairs were needed for the checking: Of the 792 TF amino acid 4-mers, 92.2% of them were found within the DNA binding domains listed in the 'PFAM 18' list downloaded from DBD [87] on 25-JAN-2010.

3.2.4 Empirical Analysis

Since the numbers of results are quite large, they are tabulated in a statistical perspective in the previous sections. This section provides readers with empirical insights into the results obtained. Comparing to the other sets, the set of TFBS5mer-TF5mer pairs shows its relative invariability to confidence level pruning. Thus it motivates us to have an in-depth empirical analysis on them. They are listed in Table 3.3.

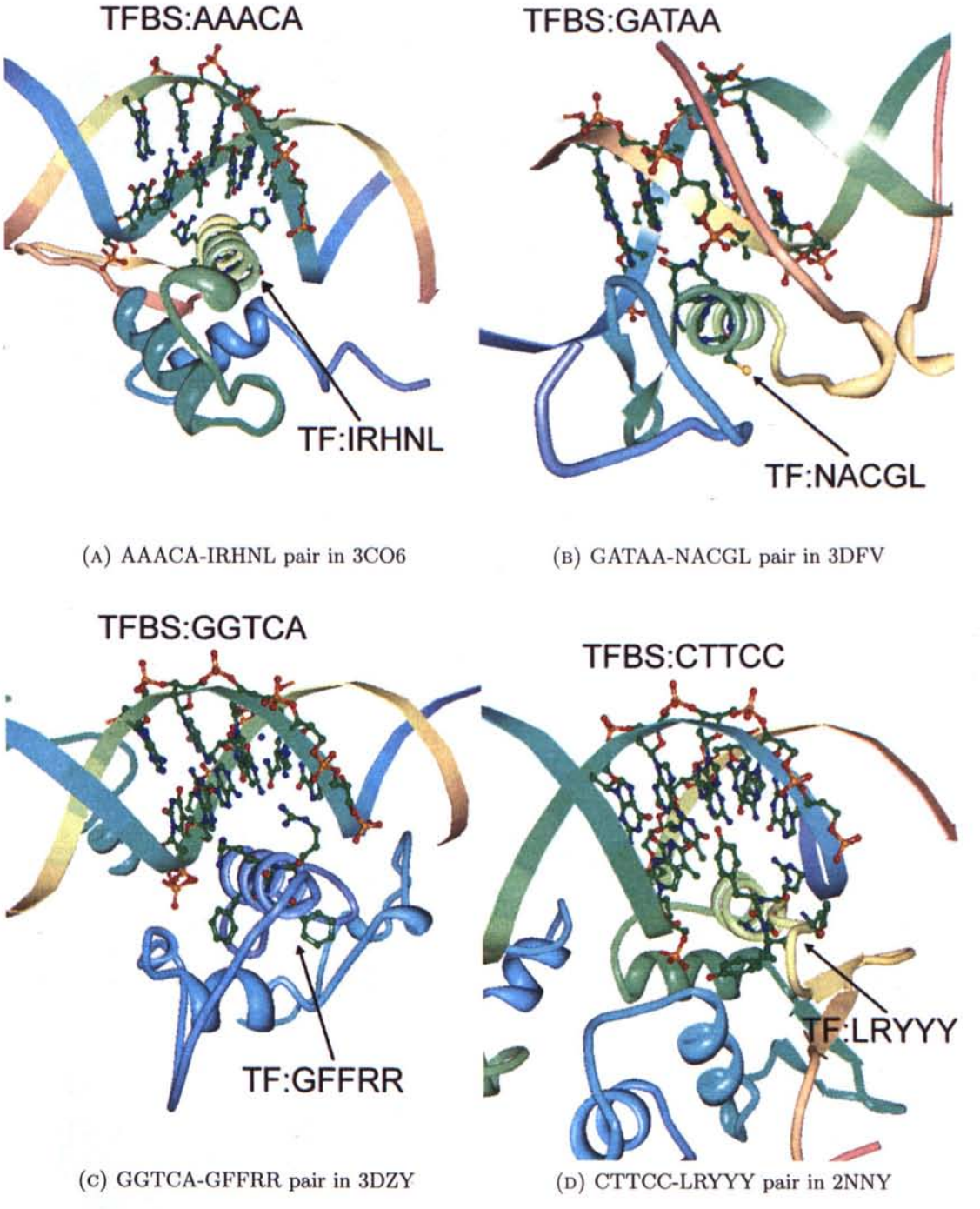
Among the 131 pairs in Table 3.3, the TFBS DNA k-mers are quite conserved. There are only 15 distinct TFBS DNA k-mers. Each TFBS DNA k-mer forms pairs with 8.73 TF amino acid k-mers on average. One of the reasons may be the specificity of DNA residue is lower in view of its alphabet size (4) as compared to the amino acid alphabet size (20).

To act as a DNA binding protein, a TF needs to provide a basic interacting surface for the recognition of major/minor grooves as well as the phosphate backbone of DNA. Therefore, we searched through the set of pairs in Table 3.3 to count the occurring frequency for each residue. Interestingly, we found that the basic residues, Lysine (50 times) and Arginine (131 times), occur at the highest frequency among 131 pairs of TFBS-TF. On the other hands, the hydrophobic residues [88] such as Isoleucine (15) and Valine (13) occur at the lowest frequency. These results suggest the potential of the TF sequences for being the binding sequences between TFs and TFBSs. On the other hand, as the nucleotides of TFBSs are somehow negatively charged, it can be deduced that their binding amino acid residues of TFs should be positively charged. Thus the occurring frequencies were further examined. Among the 131 pairs, the positively charged residues: Arginine (R) and Lysine (K) occur 131 and 50 times respectively. In contrast, the negatively charged residues Aspartic Acid (D) and Glutamic Acid (E) occur 8 and 30 times respectively. Such discrepancy supports their potential for being the binding sequences between TFs and TFBSs.

3.2.5 Experimental Analysis

This section follows the same approach in empirical analysis. The set of TFBS5mer-TF5mer pairs in Table 3.3 is selected for experimental analysis. Out of the 131 pairs, 5 of them were selected and analyzed. The first pair is GGTCA-CEGCK, which have been experimentally proved as binding sequences in [89]. The TF amino acid k-mer (CEGCK) is considered part of P-box (CEGCKG) within the DNA-binding domain of Bp-nhr-2, which is believed to bind the DNA k-mer (GGTCA). The second pair is AAACA-IRHNL mentioned in [90]. Based on the corresponding PDB entry 3CO6, it is believed that the pair was a binding pair between a TF and a TFBS as shown in Fig.3.3a. Similarly, the remaining pairs are GATAA-NACGL, GGTCA-GFFRR, and CTTCC-LRYYY. They are found as binding pairs in PDB entries 3DFV [91], 3DZY [92], and 2NNY [93] as shown in Fig.3.3b, Fig.3.3c, and Fig.3.3d respectively. The above 5 pairs reveal that the pairs generated from the proposed approach have biological evidences in literatures.

Among the previous figures, two of them (3CO6 and 2NNY) were further analyzed in terms of hydrogen bonding, which also means the specificity of the interaction between amino acids and the bases, as shown in Figures 3.4a and 3.4b. We have also highlighted the hydrogen bonds as black lines as well as the residues that make contact with the base (only predicted residues), which are the evidence of the significance and accuracy of



(A) AAACA-IRHNL pair in 3CO6

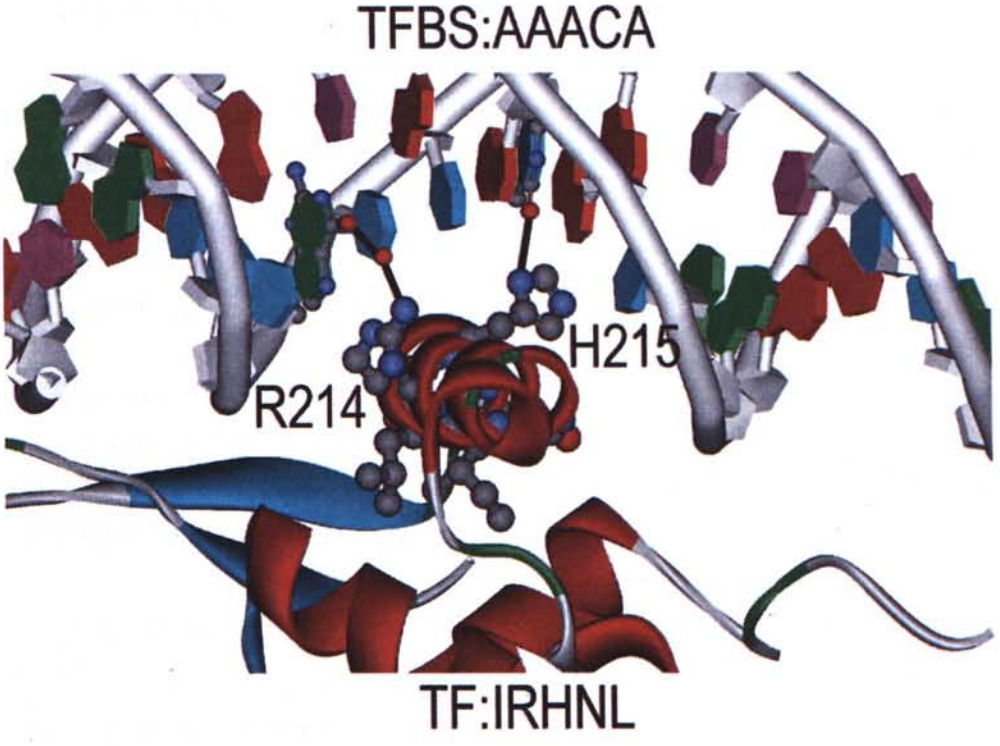
(B) GATAA-NACGL pair in 3DFV

(C) GGTCA-GFFRR pair in 3DZY

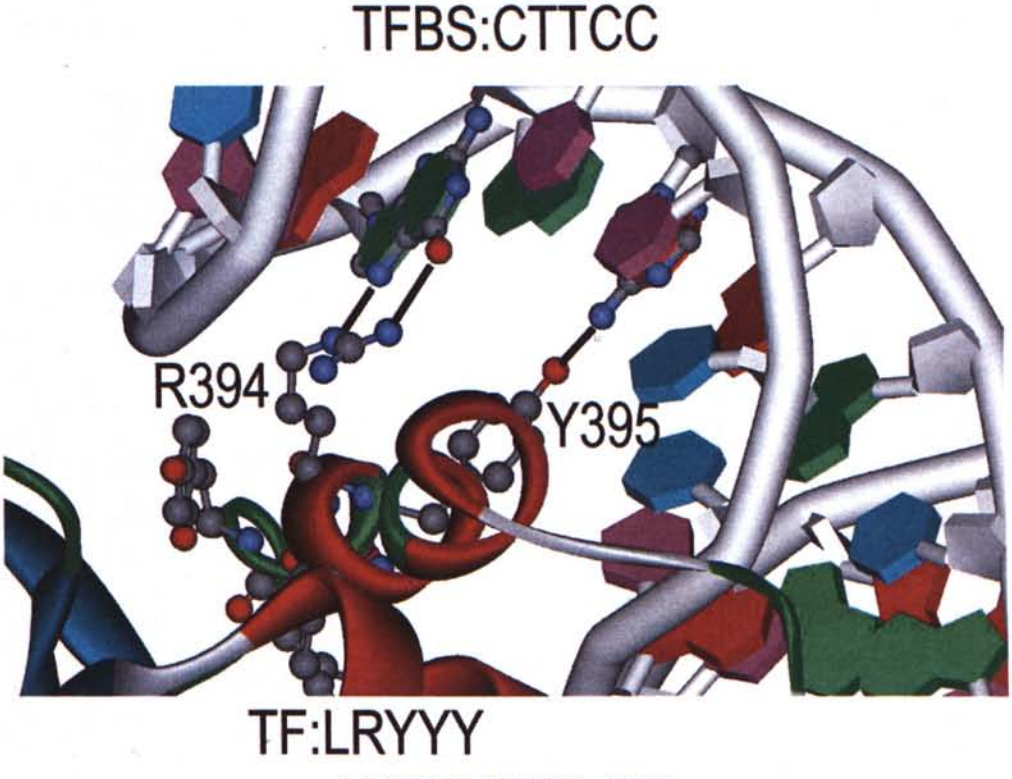
(D) CTTCC-LRYYY pair in 2NNY

FIGURE 3.3: Four representative TF-TFBS pairs are shown in ribbon diagram. The TF amino acids and TFBS nucleotides are highlighted in ball and stick format. The sequences of the TF-TFBS pairs are also labeled in the figures. The figures are generated using Protein Workshop [1]

the prediction of the TF-TFBS pairs. Nevertheless, as the proposed approach is applied on a large-scale database, such extensive and detailed analysis of all the binding core pairs discovered are not practical. Therefore, a scalable verification approach will be presented in the next section to verify the massive results generated.



(A) AAACA-IRHNL in 3CO6



(B) CTTCC-LRYYY in 2NNY

FIGURE 3.4: The interactions between the TF and TFBS of two representative pairs (a) 3CO6 and (b) 2NNY are shown. The proteins are shown in ribbon diagram with the highlighted TF amino acids in ball and stick format. The helices and strands are colored in red and cyan respectively. The amino acids that interact with the nucleotides are labeled. The hydrogen bonds are shown in dark line. The figures are generated using DS visualizer, Accelrys.

3.3 Verifications

In this section, we try to verify the discovered pairs with external data sources, in particular the 3D protein-DNA complex structures experimentally determined from PDB (Protein Data Bank). Homology modeling has also been done for further verifications.

3.3.1 Verification by PDB

PDB is selected for providing 3D Protein-DNA complex data for 3D structural verification. The PDB data were downloaded from RCSB PDB (`http://www.pdb.org`) from 16-SEP-2009 to 22-SEP-2009, where the protein-DNA complexes were selected based on the entry type list provided in `ftp://ftp.wwpdb.org/`.

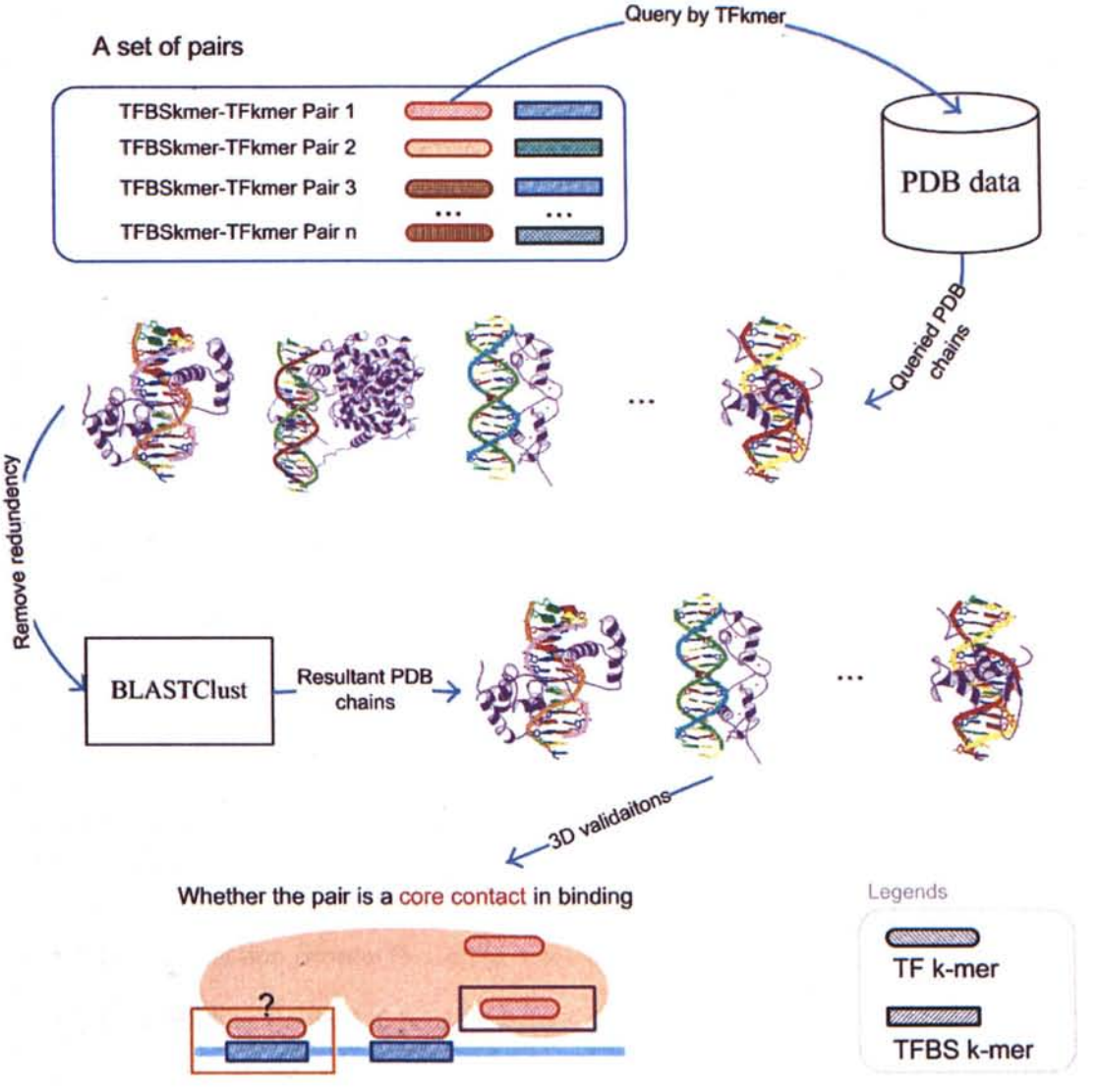


FIGURE 3.5: Flowchart of 3D Verification for each set of pairs

For each set of pairs in Supplementary Table 2, each pair is independently evaluated as shown in Fig.3.5. For each pair, its TF k-mer is used to query which PDB chain

has the TF k-mer. Once the corresponding set of PDB chains has been identified and returned, its redundancy is removed by BLASTClust using 90% sequence identity [17]. The removal is to ensure that redundant PDB chains are not double-counted. After the removal, the pair is evaluated for binding in the 3D space:

For each PDB chain queried by the TF k-mer, its protein sequence must have the TF k-mer. Thus its protein sequence is scanned to locate the sequence position of the TF k-mer. Once located, the sequence position is applied to get the corresponding 3D atomic coordinates of the TF k-mer in the PDB entry in which the PDB chain is. On the other hand, the TFBS k-mer of the pair is also scanned in the DNA sequences in the PDB entry. If the TFBS k-mer cannot be found in the DNA sequences, the PDB chain will be classified into the first category (a). Otherwise, the sequence location of the TFBS k-mer will be further applied to obtain the corresponding 3D atomic coordinates of the TFBS k-mer. As a result, the 3D atomic coordinates of the TF k-mer and the TFSB k-mer are found for the PDB chain. The following measurement is proposed to classify the PDB chain into the remaining categories (b) and (c):

- A TFBS k-mer - TF k-mer pair is considered binding for a PDB chain if and only if an atom of the TFBS k-mer and an atom of the TF k-mer are close to each other. Two atoms are considered close if and only if their distance is smaller than 3.5 angstrom. [34, 37]

With the pair evaluated in its PDB chains, its PDB chains can be classified into the following three categories:

- PDB chains only having the TF k-mer (a)
- PDB chains having both TF k-mer and TFBS k-mer
  - The pair binds together (b)
  - The pair does not bind together (c)

Thus the number of chains in each category is counted and converted into the following performance metrics:

- TFBS Prediction Score= $(b+c)/(a+b+c)$
- TFBS Binding Prediction Score = $b/(a+b+c)$
- Binding Prediction Score= $b/(b+c)$

Given the resultant PDB chains queried by a TF k-mer, TFBS Prediction Score measures the proportion of PDB chains which contain the corresponding TFBS k-mer. In other

words, it measures the backward confidence of a pair in PDB. TFBS Binding Prediction Score is a more stringent metric. It measures the proportion of PDB chains which have the corresponding TFBS k-mer binding with the queried TF k-mer. Lastly, Binding Prediction Score is the most important metric. It measures the proportion of PDB chains in which the pair is really binding. To verify the cases when $(b + c) = 0$ (i.e. the pairs do not appear in PDB), homology modeling is also performed.

For each setting, we have a set of pairs. For each pair, the above performance metrics are calculated. The overall results are averaged and summarized in Supplementary Tables 9 - 11. For each setting, we also have a set of merged pairs. For each merged pair, the above performance metrics are also calculated. The overall results are averaged and summarized in Supplementary Tables 12 - 14. Note that the most conservative calculation has been used for each performance metric for each pair. If a performance metric of a pair does not have enough PDB data for calculation, a value of zero will be given to the performance metric of the pair. For instance, the cases when $(b + c) = 0$ or $(a+b+c) = 0$. Despite the above setting, the performance metrics of the pairs still have reasonable performances. They are shown to be significant better than the maximal performance of 50 random runs in a later section.

Nevertheless, although the above metrics can capture the performance of a pair quantitatively, the most important point is to know how many generated pairs could be verified (with at least one binding evidence in PDB data ($b > 0$)). To gain more insights, the number of pairs with at least one related PDB chain ($(b + c) > 0$) are tabulated in Supplementary Tables 15 and 16. Correspondingly, the percentage of verified pairs ($\frac{Number\ of\ pairs\ with\ b>0}{Number\ of\ pairs\ with\ (b+c)>0}$) are calculated and tabulated in Supplementary Tables 17 and 18. In the tables, the percentage of verified pairs is high enough to justify that the proposed approach has produced pairs proven to be binding in PDB. For instance, the statistics for the TFBS5mer-TF5mer pairs is extracted in Table 3.4 and Fig.3.6. Among the 80 TFBS5mer-TF5mer pairs with at least one related PDB chain ($(b+c) > 0$) when the confidence level = 0.0, more than 81% of them have at least one binding evidence ($b > 0$).

The TFBS-TF pairs that we found to have binding evidences in the PDB show typical structural features of DNA-protein interactions. Such features include the "recognition helix" of the DNA-binding protein making base contacts in the major groove and direct hydrogen bonds between the side chains and the bases. These interactions play the crucial role in the DNA recognition and site specific binding respectively [94]. Interestingly, the nucleotides of TFBS are located in the major groove of the DNA, which are close to, and make contacts with the amino acids of the "recognition helix" of the TF (as for example shown in Figure 3.3).

TABLE 3.4: Number of the TFBS5mer-TF5mer pairs verified across different confidence levels. ($N_{related}$: Number of the TFBS5mer-TF5mer pairs with at least one related PDB chain ($(b+c)>0$), $N_{verified}$: Number of the TFBS5mer-TF5mer pairs with at least one PDB chain as a binding evidence ($(b)>0$), $M_{related}$: Number of the TFBS5mer-TF5mer merged pairs with at least one related PDB chain ($(b+c)>0$), $M_{verified}$: Number of the TFBS5mer-TF5mer merged pairs with at least one PDB chain as a binding evidence ($(b)>0$))

Confidence	$N_{related}$	$N_{verified}$	$M_{related}$	$M_{verified}$
0.0	80	65	19	16
0.1	80	65	19	16
0.2	71	59	15	13
0.3	50	44	15	13
0.4	32	28	12	11
0.5	19	17	7	6
0.6	9	9	5	5
0.7	2	2	1	1
0.8	0	0	0	0
0.9	0	0	0	0
1.0	0	0	0	0

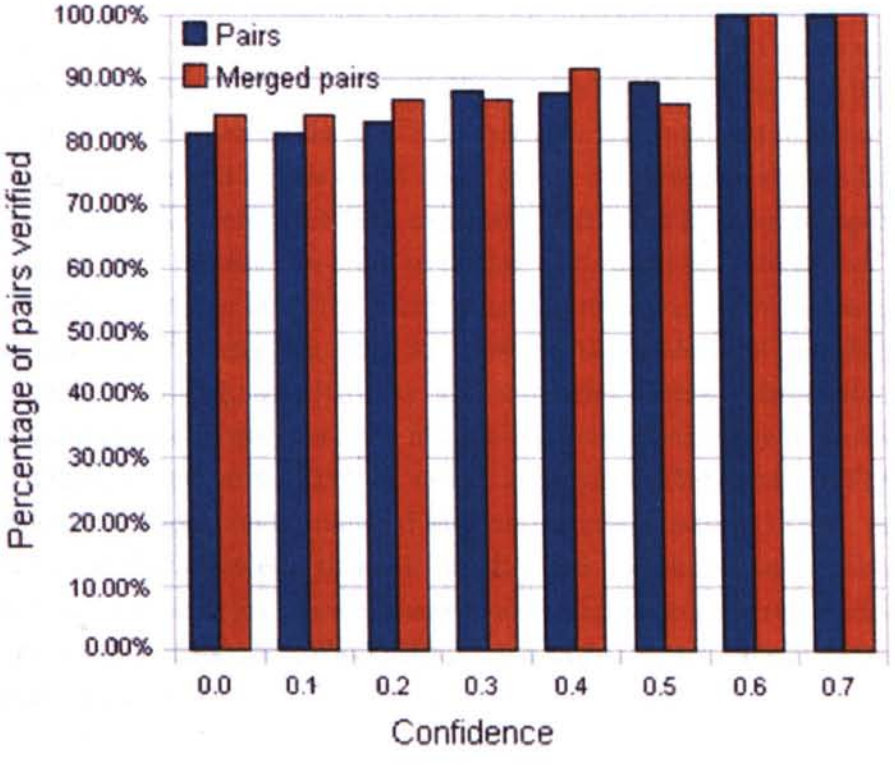


FIGURE 3.6: Percentage of the TFBS5mer-TF5mer pairs verified across different confidence levels

The verification is considered satisfactory since those pairs not found in PDB ($(b+c) = 0$) may be un-annotated discovery as shown in the following verification by homology modeling.

3.3.2 Verification by Homology Modeling

Regarding the pairs without any related PDB chain ($(b+c) = 0$), there is no PDB data for us to verify them. Thus we have taken the most conservative approach to assign zero to their performance metrics in the aforementioned evaluations. Nevertheless, we believe that most of those pairs are true and our approach can be used as an effective protein-DNA binding discovery tool. Thus 6 TFBS5mer-TF5mer pairs were taken and merged. The resultant pair ACGTG-SNRESARRSR was analyzed by homology modeling as follows:

The model of DNA-protein complex was built by homology modeling (INSIGHT II, MSI) based on the structure of the GCN4-DNA complex (1YSA) [95]. Briefly, three amino acids (R234S, T236R, and A238S) and two nucleotides (T29C and A31T) were mutated in the original structure. The side chains of the mutated amino acids were chosen from the rotamer database and examined using the Ramachandran plots to prevent any steric effect. The interactions between the amino acids and the nucleotides were searched based on the distance of the hydrogen bond.

As shown in Fig.3.7, we found that the pair ACGTG-SNRESARRSR exists in plant as the basic leucine-zipper (bZIP) transcription factor which binds to G-box binding factors (GBF) of DNA [96]. Moreover, the ACGTG sequence is the consensus sequence which is defined as G-box core and locates at the major groove of the double strands DNA. It is believed that the G-box core is the DNA sequence of GBF that provides the specificity of the binding to bZIP proteins. In order to further understand the interactions between the TF-TFBS, we built a model by using homology modeling based on the structure of GCN4-DNA (1YSA) complex [95]. As shown in the model, the protein helix fits into the major groove of the DNA very well and forms extensive interactions (black lines) between the amino acids and the nucleotides. Interestingly, the mutations of the protein (R234S, T236R, and A238S) as well as nucleotides (T29C and A31T) increases the number of hydrogen bonds compared with the original structure (1YSA), suggesting the binding specificity between this pair of TF-TFBS. In conclusion, we believe that the protein-DNA binding sequence patterns found using association rule mining on the large-scale database reveal real TF-TFBS pairs in physiological relevant situation and this method could guide us to discover new and undescribed TF-TFBS pairs in the future.

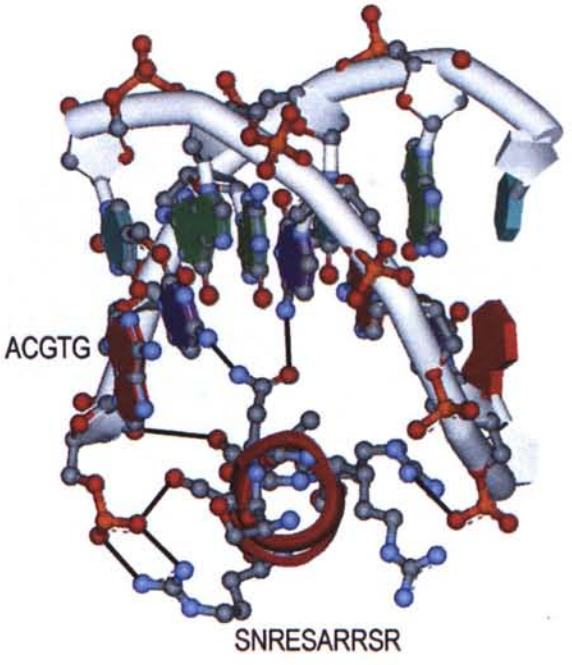
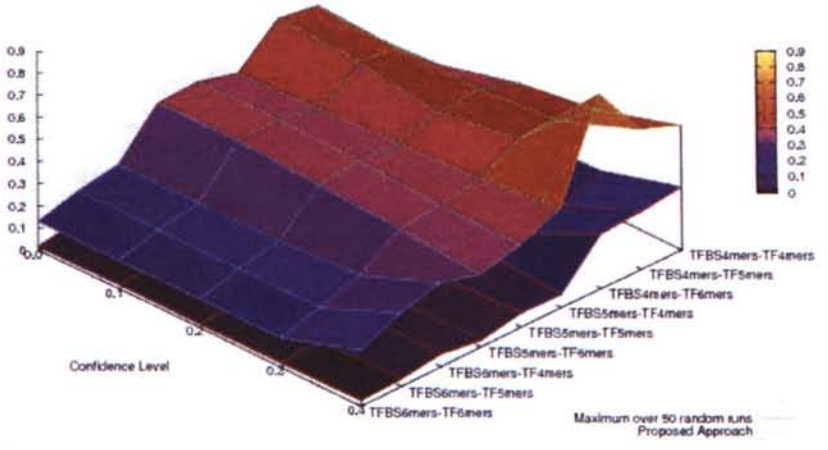


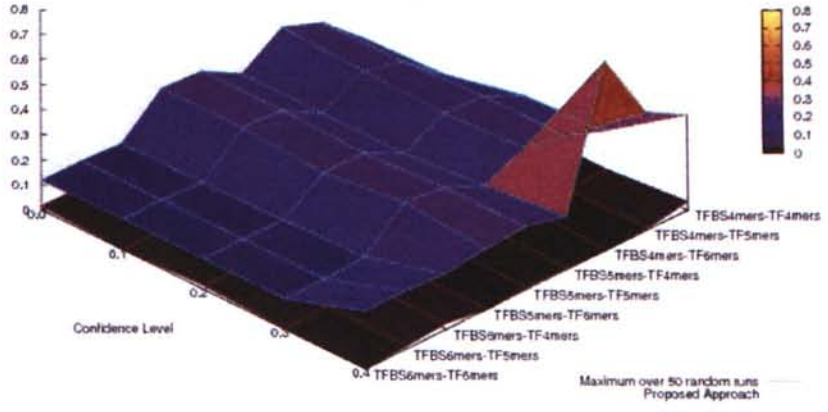
FIGURE 3.7: The pair ACGTG-SNRESARRSR using homology modeling

3.3.3 Verification by Random Analysis

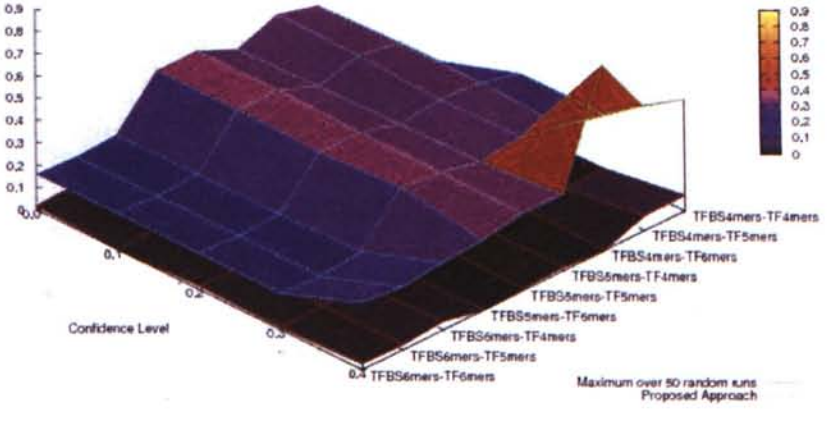
For each set of pairs in Supplementary Table 1, we use a random process to generate a random set with the same number of pairs. Within a random set, its pairs were randomly sampled from all the combinations of the k-mers used in the proposed approach. 50 random runs were performed. The maximal performance metrics of the 50 random runs are summarized in Supplementary Tables 19 - 21. In a comparison to the proposed approach, their performance has been depicted in Figure 3.8 and Figure 3.9. It can be observed that the performance of the proposed approach is significantly better than the best one of the 50 random runs. For instance, the Binding Prediction Score of the 131 TFBS 5mer-TF 5mer pairs generated is 0.36 ± 0.39 on average whereas the maximal Binding Prediction Score over 50 random runs is only 0.00509 ± 0.06492 on average. Similar observation can also be drawn for their merged pairs in Supplementary Tables 22 - 24. It can be concluded that the performance of the proposed approach is very unlikely to happen purely by chance in PDB.



(A) TFBS Prediction Score

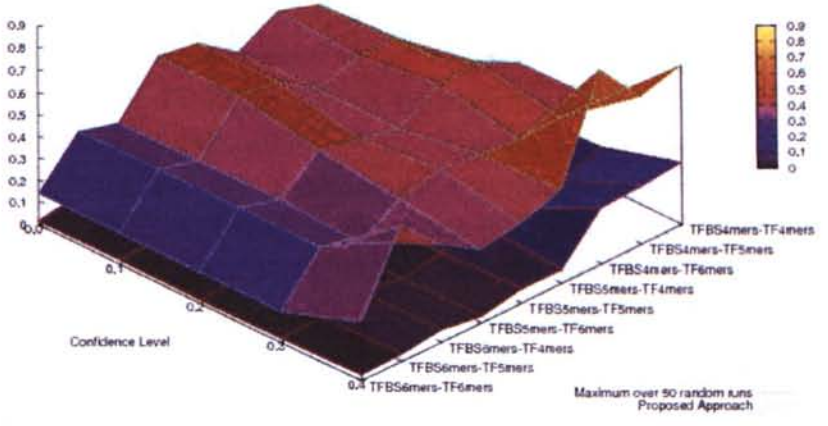


(B) TFBS Binding Prediction Score

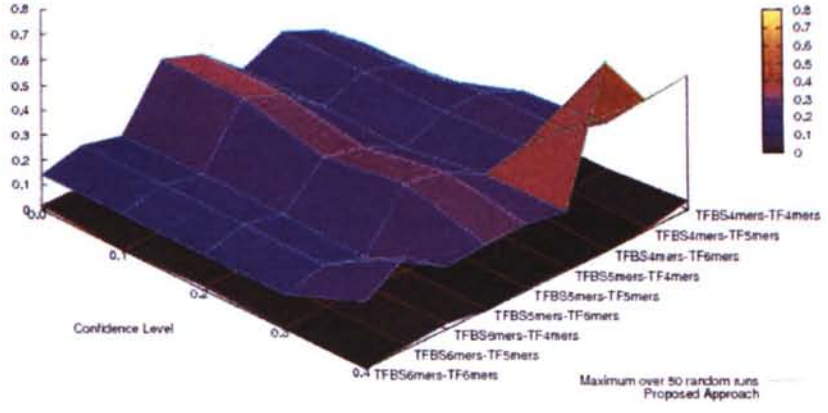


(C) Binding Prediction Score

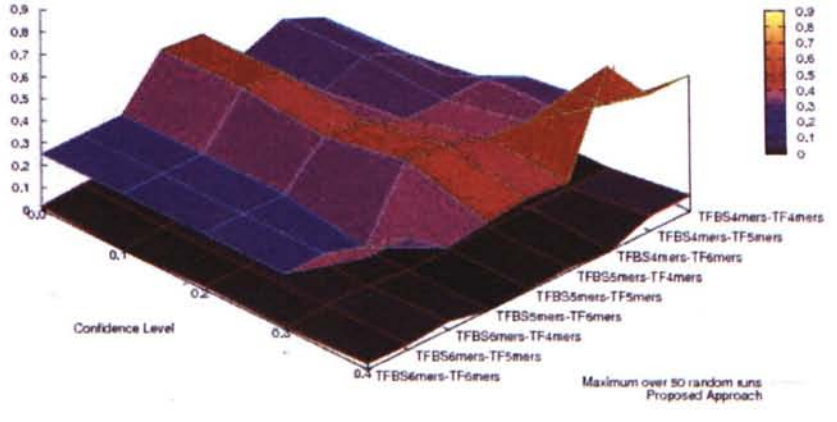
FIGURE 3.8: Performance Comparison for PDB verifications



(A) TFBS Prediction Score (Merged Pairs)



(B) TFBS Binding Prediction Score (Merged Pairs)



(C) Binding Prediction Score (Merged Pairs)

FIGURE 3.9: Performance Comparison for PDB verifications (Merged Pairs)

3.4 Discussion

We have proposed a framework based on association rule mining with Apriori algorithm to discover associated TF-TFBS binding sequence patterns in the most explicit and interpretable form from TRANSFAC. With downward closure property, the algorithm guarantees the exact and optimal performance to generate all frequent TFBS k-mer TF k-mer pairs from TRANSFAC. The approach relies merely on sequence information without any prior knowledge in TF binding domains nor protein-DNA 3D structure data. From comprehensive evaluations, statistics of the discovered patterns are shown to reflect meaningful binding characteristics. According to external literatures, PDB data, and homology modeling, a good number of TF-TFBS binding patterns discovered have been verified by experiments and annotations. They exhibit atomic-level bindings between the respective TF binding domains and specific nucleotides of the TFBS from experimentally determined protein-DNA 3D structures. In fact, most of the pairs discovered are actually the binding cores from the TF binding domains and TFBS respectively.

The proposed approach has great potential for discovering intuitive and interpretable rules of TF-TFBS binding mechanisms. Such rules are able to reveal TF binding domains, detailed interactions between amino acids and nucleotides, accurate TFBS sequence motifs, and help better understanding and deciphering of protein-DNA interactions. It also offers strategic help to reduce the labor and costs involved in wet-lab experiments. With increasing computational power and more sophisticated mining approaches, the proposed methodology can be further improved for discovering more intriguing TF-TFBS binding patterns and rules.

Chapter 4

Designing Evolutionary Algorithms for Multimodal Optimization

4.1 Introduction

Since genetic algorithm was proposed by John Holland [24] in the early 1970s, evolutionary algorithm has emerged as a popular research field. Researchers around the world have been digging into this field and exploring the power of evolutionary algorithms. In particular, its function optimization capability was highlighted [32] because of its high adaptability to different function landscapes, to which we cannot apply traditional optimization techniques.

Real world problems always have different solutions. For instance, in the varied-line-spacing holographic grating design problem, optical engineers need to tune the recording parameters to get as many optimal solutions as possible for multiple trials in the design problem. Because the design constraints are too difficult to be expressed and solved in mathematical forms [97]. Unfortunately, most traditional optimization techniques focus on solving for a single optimal solution. They needs to be applied several times; yet all solutions are not guaranteed to be found. Thus multimodal optimization problem was proposed. In this problem, we are interested in not only a single optimal point, but also the others. Given an objective function, an algorithm is expected to find all optimal points in a single run. With strongly parallel search capability, evolutionary algorithms are shown to be particularly effective in solving this type of problems [32].

The work by De Jong [98] is one of the first known attempts to solve multimodal optimization problems by an evolutionary algorithm. He introduced the crowding technique

to increase the chance for locating multiple optima. In the crowding technique, an offspring replaces the parent which is most similar to the offspring itself. Such a strategy can preserve the diversity and maintain different types of individuals in a run. Twelve years later, Goldberg and Richardson [60] proposed a fitness-sharing niching technique as a diversity preserving strategy to solve multimodal optimization problems. He proposed a shared fitness function, instead of an absolute fitness function, to evaluate the fitness of an individual in order to favor the growth of the individuals which are distinct to others. With this technique, a population can be prevented from the domination of a particular type of individuals. Since then, many researchers started to explore different ways to deal with the problems. These methods include: species conserving [99], crowding [98, 100], elitism [101], differential evolution [102], clearing [103], repeated iterations [104] and island model [105]. Though different methods were proposed in the past, they were all based on the same fundamental idea. It is to strike an optimal balance between convergence and diversity of evolutionary algorithm in order to locate all optima (global and local), as defined as follows:

4.2 Problem Definition

The problem definition depends on the type of optimization (minimization or maximization). They are similar in principle and defined as follows:

4.2.1 Minimization

In this problem, given f:$\mathbb{X} \rightarrow \mathbb{R}$, we would like to find all global and local minimums of f in a single run.

Definition 4.1 Local Minimum [106]: A (local) minimum $\hat{x}_l \in \mathbb{X}$ of one (objective) function f:$\mathbb{X} \rightarrow \mathbb{R}$ is an input element with $f(\hat{x}_l) \leq f(x)$ for all x neighboring $\hat{x}_l$. If $\mathbb{X} \in \mathbb{R}^N$, we can write: $\forall \hat{x}_l \; \exists \epsilon > 0 : \; f(\hat{x}_l) \leq f(x) \; \forall x \in \mathbb{X}, |x - \hat{x}_l| < \epsilon$.

Definition 4.2 Global Minimum [106]: A global minimum $\hat{x}_l \in \mathbb{X}$ of one (objective) function f:$\mathbb{X} \rightarrow \mathbb{R}$ is an input element with $f(\hat{x}_l) \leq f(x) \; \forall x \in \mathbb{X}$.

4.2.2 Maximization

In this problem, given f:$\mathbb{X} \rightarrow \mathbb{R}$, we would like to find all global and local maximums of f in a single run.

Definition 4.3 Local Maximum [106]: A (local) maximum $\hat{x}_l \in \mathbb{X}$ of one (objective)

function f:$\mathbb{X} \rightarrow \mathbb{R}$ is an input element with $f(\hat{x}_l) \geq f(x)$ for all x neighboring $\hat{x}_l$. If $\mathbb{X} \in \mathbb{R}^N$, we can write: $\forall \hat{x}_l \; \exists \epsilon > 0 : \; f(\hat{x}_l) \geq f(x) \; \forall x \in \mathbb{X}, |x - \hat{x}_l| < \epsilon$.

Definition 4.4 Global Maximum [106]: A global maximum $\hat{x}_l \in \mathbb{X}$ of one (objective) function f:$\mathbb{X} \rightarrow \mathbb{R}$ is an input element with $f(\hat{x}_l) \geq f(x) \; \forall x \in \mathbb{X}$.

4.3 An Evolutionary Algorithm with Species-specific Explosion for Multimodal Optimization

The species conserving technique for multimodal optimization was proposed by Li et al. [99]. It was claimed that the technique was considered as an effective and efficient method for inducing niching behavior into GAs. However, in our experiments, we find that the performance of the technique still has space for improvement. It always suffers from genetic drifts though each species is conserved with one individual. The result of the comparison test conducted by Singh et al. [107] also reveals that the species conserving technique performs the worst among the algorithms tested. As a result, we propose a novel algorithm to remedy the species conserving technique in this section.

The novel algorithm is called Evolutionary Algorithm with Species-specific Explosion (EASE) for multimodal optimization. EASE is built on the Species Conserving Genetic Algorithm (SCGA), and the design is improved in several ways. In particular, it not only identifies species seeds, but also exploits the species seeds to create multiple mutated copies in order to further converge to the respective optimum for each species. Experiments were conducted to compare EASE and SCGA on eight benchmark functions.

4.3.1 Background

4.3.1.1 Species Conserving Genetic Algorithm

Species conserving genetic algorithm (SCGA) [99] is a technique for evolving parallel subpopulations for multimodal optimization. Before each generation starts, the algorithm selects a set of species seeds which can bypass the subsequent procedures and be saved into the next generation. The algorithm then divides a population into several species based on a dissimilarity measure. The fittest individual is selected as the species seed for each species. After the identification of species seeds, the population undergoes the usual genetic algorithm operations: selection, crossover and mutation. As the operations may remove the survival of less fit species, the saved species seeds are copied back to the population at the end of each generation. The whole structure of SCGA is outlined in Appendix.

To determine the species seeds in a population, the algorithm first sorts the population in a decreasing fitness order. Once sorted, it picks up the fittest individual as the first species seed and forms a species region around it. The next fittest individual is tested whether it is located in a species region. If not, it is selected as a species seed and another species region is created around it. Otherwise, it is not selected. Similar operations are applied to the remaining individuals, which are subsequently checked against all existing species seeds.

To copy the species seeds back to the population after the genetic operations have been executed, the algorithms need to scan all the individuals in the current population and identify to which species they belong. Once it is identified, the algorithm replaces the worst individual (lowest fitness) with the species seed in a species. If no individuals can be found in a species for replacement, the algorithm replaces the worst and un-replaced individual in the whole population. In short, the main idea is to preserve the population diversity by preserving the fittest individual for each species.

4.3.2 Evolutionary Algorithm with Species-specific Explosion

Evolutionary Algorithm with Species-specific Explosion (EASE) is an evolutionary algorithm which identifies and exploits species seeds to locate global and local optima. There are two stages in the algorithm: **Exploration Stage** and **Species-specific Stage**. The exploration stage targets for roughly locating all global and local optima. It not only undergoes normal genetic operations: selection and crossover, but also involves the addition of randomly generated individuals for preserving the diversity. On the other hand, the species-specific stage targets for gently locating the optimum for each species. Species-specific genetic operations are applied. Only the individuals within the same species are allowed to perform selection and crossover to each other. No inter-species selection and crossover are allowed. Such a strategy is to provide more chances for each species to converge to its respective optimum, with the trade-off that diversity is no longer preserved. To have a better global picture for locating optima, EASE starts with the exploration stage. It will switch to the species-specific stage only after the stage switching condition is satisfied. No matter in which stage, a local operation called **Species-specific Explosion** is always executed so as to help species to climb and converge to its corresponding optimum. The whole structure of EASE is shown in Algorithm 4.

4.3.2.1 Species Identification

To determine species in a population, we adopt the dissimilarity measurement proposed in Goldberg and Richardson [60] and Li et al. [99]. The dissimilarity between two

[3]It involves Survival Selection if the generation is overlapping.

Algorithm 4 Evolutionary Algorithm with Species-specific Explosion

$G(t)$: Generation at time t
X_s: A set storing species seeds
E_s: A set storing species-specific exploded individuals
pop_size: Initial population size
K: A real number over the interval $[0, 1]$
SS: Species Specific stage switching parameter
$EMSS$: Expected Mutation Step Size

$t \leftarrow 0$;
$SS \leftarrow false$;
$EMSS \leftarrow$ mutation probability $\times$ mutation step size;
$pastIndividuals \leftarrow \emptyset$;
Initialize $G(t)$;
Evaluate $G(t)$;
while not termination condition **do**
  if $SS = false$ **then**
    Select $G(t+1)$;
    Crossover $G(t+1)$;
  else
    Species-specific Select $G(t+1)$;
    Species-specific Crossover $G(t+1)$;
  end if
  Mutate $G(t+1)$;
  Evaluate $G(t+1)$[3];
  $X_s \leftarrow$ IDENTIFYSPECIESSEEDS($G(t+1)$);
  DELTAEVAL($X_s, pastIndividuals$);
  if $SS = false$ **then**
    $SS \leftarrow$ ISSPECIESSPECIFIC($X_s, EMSS$);
  end if
  $E_s \leftarrow$ SPECIESSPECIFICEXPLOSION($G(t+1), X_s, K, SS$);
  $pastIndividuals \leftarrow X_s \cup E_s$;
  $G(t+1) \leftarrow X_s \cup E_s$;
  if $SS = false$ **then**
    Fills $G(t+1)$ with randomly generated individuals
    to reach *pop_size*;
  end if
  $t \leftarrow t+1$;
end while
Identify species seeds X_s;
Identify global optima;

individuals are based on their Euclidean distance. The smaller the distance, the more similar they are:

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})^2}$$

where x_i and x_j are two individuals, which are n-dimensional vectors $[x_{i0}, x_{i1}, ..., x_{in}]$ and $[x_{j0}, x_{j1}, ..., x_{jn}]$ respectively.

Each species is a subset of population. The fittest individual within a species is chosen as the species seed. The region around a species seed forms its corresponding species region. All individuals are classified as the same species if it is within the species distance (r_s) from the species seed. Petrowski [103] and Li et al. [99] proposed an algorithm to identify species seeds. The algorithm first sorts the population in a decreasing fitness order. Then it picks up the individual with the highest fitness as the first species seed and forms a species region around it. All individuals within r_s distance from the species seed are classified as the same species as that of the seed. For the next individual, it is checked whether it is within r_s distance from the species seed. If not, it is selected as another species seed. Similar operations are applied to the remaining individuals. Each individual is tested on whether it lies in others' species regions. If not, it is selected as a species seed. Otherwise, it is not selected. The main idea is to pick up the fittest individuals as the species seed for each species. The algorithm is shown in Algorithm 5.

Algorithm 5 Identify Species Seeds

```
procedure IDENTIFYSPECIESSEEDS(G)
    Sort G in decreasing fitness values;
    X_s ← ∅;
    while not reaching the end of G do
        Get best un-scanned individual i_s from G;
        found ← false;
        for ∀x ∈ X_s do
            if d(x, i_s) ≤ r_s then
                found ← true;
                break;
            end if
        end for
        if notfound then
            X_s ← X_s ∪ i_s;
        end if
    end while
    return X_s;
end procedure
```

4.3.2.2 Species Seed Delta Evaluation

After we have identified all species seeds in the population, we perform delta evaluation to record the recent step changes that can increase fitness for each species seed. For each species seed, we pick up the fittest individual of the same species in the previous generation, under the constraint that its fitness is lower than that of the species seed itself. By doing so, we can select the individual which is most likely the ancestor of a species seed in the previous generation. We call this individual as the Likelihood

Ancestor(LA). If we can pick up the corresponding LA for a species seed, we store the value difference between the genome of LA and the genome of the species seed into the array *delta* of the species seed. Thus the array *delta* of a species seed serves as a memory recording the last known step sizes, which improved the species seed itself. The algorithm is shown in Algorithm 6. (All elements of *delta* are initialized to the mutation step size at the beginning)

Algorithm 6 Species Seed Delta Evaluation

dim: The maximum dimension
$x.value\,[i]$: The genome value of x at dimension i
$LA.value\,[i]$: The genome value of LA at dimension i

procedure DELTAEVAL($X_s, pastIndividuals$)
  for $\forall x \in X_s$ **do**
    $LA \leftarrow$ the individual $\in pastIndividuals$
    with the highest fitness in the same species
    where its fitness is lower than that of x and $x \neq LA$;
    if $LA \neq null$ **then**
      for i from 1 to dim **do**
        $x.delta\,[i] \leftarrow x.value\,[i] - LA.value\,[i]$;
      end for
    end if
  end for
end procedure

4.3.2.3 Stage Switching Condition

To ensure a proper condition for switching from the exploration stage to the species-specific stage, we propose using the expected mutation step size ($EMSS$) as a measure for controlling the switching:

$$EMSS = p_m \times r_m$$

where p_m is the mutation probability and r_m is the mutation step size.

For each species seed, we scan its array *delta* to check whether its element exceeds $EMSS$. If there does not exist an element which exceeds $EMSS$, the switching condition is satisfied. The algorithm will switch to the species-specific stage. Otherwise, the algorithm will remain in the exploration stage. The rationale behind the checking condition is that $EMSS$ can give us an expected value for measuring the mutation ability of the algorithm. It can serve as a measurement to assess the ability of the algorithm to jump from one region to another region by just using mutation. Thus if all the elements of the arrays *delta* of all species seeds do not exceed the $EMSS$, it is reasonable to deduce that the fitness improvement steps for all species in the subsequent generations are most likely smaller than the $EMSS$. All fitness improvement steps can be completed by merely using mutations, but not inter-species crossovers. Hence inter-species crossovers

are no longer needed. Species-specific stage should be launched. The algorithm is shown in Algorithm 7.

Algorithm 7 Stage Switching Condition

dim: The maximum dimension

```
procedure isSpeciesSpecific(X_s, EMSS)
    SS ← true;
    for ∀x ∈ X_s do
        for i from 1 to dim do
            if x.delta[i] ≥ EMSS then
                SS ← false;
            end if
        end for
    end for
    return SS;
end procedure
```

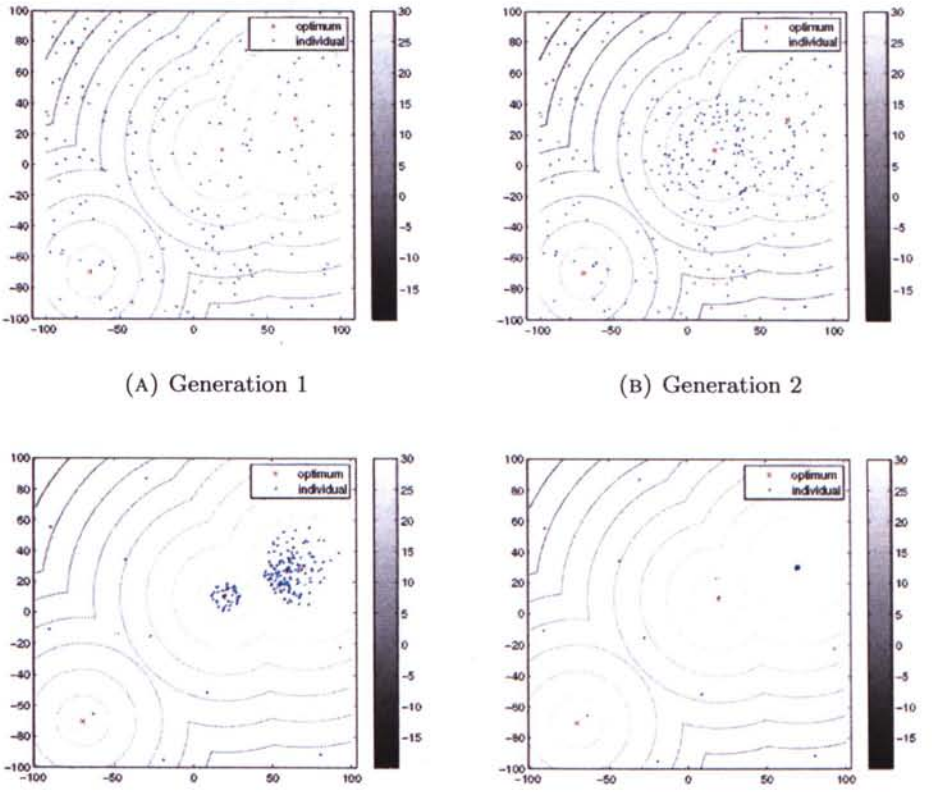
4.3.2.4 Species-specific Explosion

In SCGA, Li et al. [99] proposed conserving one individual for each species. However, just one individual for each species is not enough for the algorithm to well-conserve and nurture the species. In a run of SCGA, it is often the case that the algorithm does conserve species with low fitness values, but they are present in a small proportion. Once they form new offspring, their offspring are often removed quickly in subsequent generations due to their low fitness values. Thus most individuals are always of the species with high fitness values. An example is depicted in Figure 4.1. In the example, we can observe that the individuals gradually converge to the three optima fitness-proportionally. Though different species are preserved with an individual as the species seed, it cannot converge to the local optimum located at the left-bottom corner due to the relatively low fitness values there. Merely SCGA itself actually cannot provide enough indiscriminate condition for species to evolve and converge to its respective optimum in each run. Hence we propose a local operation called **Species-specific Explosion** to remedy the convergences in this paper.

Species-specific explosion is the local operation in which we create multiple copies for each species seed and mutate them. To start this local operation, the algorithm needs to determine two parameters:

1. How many copies should be created for each species seed?
2. What is the mutation step size for each species seed?

For the first question, we propose using the ratio of individuals in the same species to the current population to determine the number of copies to be created. The details are



(A) Generation 1

(B) Generation 2

(C) Generation 4

(D) Generation 10

FIGURE 4.1: A snapshot of SCGA in a run on Problem Peaks1 - generation 1,2,4,10.

given in the section 4.3.2.5. For the second question, we propose using the array *delta* of species seed as the corresponding mutation step size. Recall that the array *delta* of species seed saves the step size values which were known to improve the species seed itself in the previous generation, it can be used for approximating how far the species seed should mutate to have a better fitness in the current generation. Hence we choose to use the array *delta* as the mutation step size of the species-specific explosion operation. Once the two parameters are calculated, the algorithm starts to check whether the species seed is present in the previous generation. If it is present, the algorithm will "explode" it, which means creating multiple copies and mutating them. Otherwise, no actions will be executed. The rationale behind the checking is to ensure that the species seed to be exploded is a stable species seed. Hence we require the species seed at least survive through one generation to be eligible for the explosion. Alternatively, if the current stage is species-specific stage, the above checking is overridden. All species seeds are eligible for the explosions, in order to provide all species an indiscriminate condition to evolve in this stage. The algorithm is shown in Algorithm 8.

Algorithm 8 Species Specific Explosion

```
procedure SPECIESSPECIFICEXPLOSION(G, X_s, K, SS)
    E_s ← ∅;
    WEIGHTSEVAL(X_s, G);
    for ∀x ∈ X_s do
        if x is present in previous generation
      or SS = true then
            size ← x.weight × K × pop_size;
            E_s ← E_s ∪ EXPLODE(x, size);
        end if
    end for
    return E_s;
end procedure

procedure EXPLODE(x, size)
    Exploded_s ← ∅;
    for i from 1 to size do
        temp ← Copy of x;
        Mutate temp with step size x.delta;
        Exploded_s ← Exploded_s ∪ temp;
    end for
    return Exploded_s;
end procedure
```

4.3.2.5 Calculate Explosion Weights

Before an explosion, we need to determine the explosion weight for each species seed. The explosion weight is defined over $[0, 1]$. It is a scaling factor to determine the number of mutated copies that a species seed can create during the species-specific explosion process. In EASE, the rationale behind is to encourage a species to create more mutated copies if the species has less individuals in the current population. Hence the explosion weight of a species seed is derived from the ratio of individuals in the same species to the current population. The larger the ratio, the smaller is the explosion weight and vice versa. The algorithm is shown in Algorithm 9. Each explosion weight is normalized at the end so that the sum of all the explosion weights is limited to 1, in order to avoid the total number of the mutated copies of all species seeds exceeding the predefined value $K \times pop\_size$.

4.3.3 Experiments

We implemented EASE and SCGA using Sun's Java programming language. Its development is based on the EC4 framework provided in Kenneth De Jong's book [57]. Experiments to compare the performance between EASE and SCGA were conducted on eight benchmark functions as shown below:

Algorithm 9 Calculate the explosion weight for each species seed

```
procedure WEIGHTSEVAL(X_s, G)
    total ← 0;
    for ∀x ∈ X_s do
        x.weight ← population size - number of individuals
        in the same species in the current population;
        total ← total + x.weight;
    end for
    for ∀x ∈ X_s do
        x.weight ← x.weight/total;
    end for
end procedure
```

- F1: Deb's 1st function [108]
- F2: Himmelblau function [104]
- F3: Six-hump Camel Back function [109]
- F4: Branin function [109]
- F5: Rosenbrock function [110]
- F6: PP1 [111]
- F7: PP3 [111]
- F8: PP4 [111]

4.3.3.1 Performance measurement

For multimodal optimization, there are several performance metrics proposed previously [112]. The focuses of this paper are on (1) the ability of the algorithms to locate the optima and (2) the accuracy of the optima found by the algorithms. Hence we use the peak ratio and the average minimum distance to the real optima [111] as the performance metrics.

- A peak is considered found when there exists an individual which is within 0.5 Euclidean distance to the peak in the last population. Thus the peak ratio is calculated using the following formula:

$$PeakRatio = \frac{Number\ of\ peaks\ found}{Total\ number\ of\ peaks}$$

- The average minimum distance to the real optima is calculated using the following formula:

$$D = \frac{\sum_{i=1}^{n} \min_{indiv \in pop} d(peak, indiv)}{n}$$

TABLE 4.1: Parameter setting of EASE for all benchmarks

Parameter	Setting
Mutation Type	Gaussian [57]
Mutation Formula in explosion	$NewValue = OldValue + 2 \times StepSize \times U$ where U is a normally distributed real value with mean 0.0 and standard deviation 1.0
K	0.4

TABLE 4.2: Common parameter setting of EASE and SCGA for all benchmarks

Parameter	Setting
Population Initialization	Random
Generation Type	Overlapping [57]
Parent Selection	Fitness Proportional
Survival Selection	Truncation [57]
Representation	Sun's Java Double (double-precision 64-bit IEEE 754 floating point)
Mutation Type	Gaussian [57]
Mutation Formula	$NewValue = OldValue + 1.3 \times StepSize \times U$ where U is a normally distributed real value with mean 0.0 and standard deviation 1.0
Mutation Probability	0.2
Mutation Step Size	0.1
Crossover Type	Intermediate Recombination [99]
Crossover Formula	$Offspring = \frac{Parent_1 + Parent_2}{2}$
Crossover Probability	1
Random Seed	12345
Implementation	Sun's Java programming language

where n is the number of peaks, *indiv* denotes an individual and *pop* denotes a population of individuals.

In the following sections, all algorithms were run up to maximum 40000 fitness function evaluations. The above performance metrics were obtained by taking the average and standard deviation of 50 runs.

4.3.3.2 Parameter settings

The parameter setting of EASE for all benchmarks is shown in Table 4.1. The common parameter setting of EASE and SCGA for all benchmarks is shown in Table 4.2. The common parameter setting of EASE and SCGA for different benchmarks is shown in Table 4.3. All the common parameter settings of EASE are exactly the same as SCGA for fair comparisons. The selection method of the species distance parameters is based on the suggestions in [99].

4.3.3.3 Results

Table 4.4 shows the experimental results for the comparison of EASE and SCGA. It can be observed that EASE outperformed SCGA in all the benchmark functions. EASE does improve SCGA's performance no matter in the ability to locate optima or the accuracy

[2]Using the terms in [99], the species distance $(r_s) = \sigma_s/2$

TABLE 4.3: Common parameter setting of EASE and SCGA for different benchmarks

Benchmark	Population Size	Species Distance^2
F1	100	0.01
F2	100	3
F3	100	0.5
F4	100	6
F5	100	10
F6	200	50
F7	200	3
F8	200	3

of the optima found. Besides, UN is a canonical evolutionary algorithm in the EC4 framework. Its results can be regarded as the baseline results without any multimodal optimization techniques.

4.3.4 Conclusion

A new evolutionary algorithm for multimodal optimization called Evolutionary Algorithm with Species-specific Explosion(EASE) is proposed. EASE is an algorithm to remedy SCGA by exploding species seeds for locating optima.

EASE is divided into two stages: **Exploration Stage** and **Species-specific Stage**. EASE starts with the exploration stage. Once the stage switching condition is satisfied, it will be changed to species-specific stage. Throughout the two stages, a local operation: **Species-specific Explosion** is applied so as to help each species to converge to its respective optimum.

The experimental results show that EASE improves SCGA for locating optima (global and local), in terms of peak ratio and accuracy.

TABLE 4.4: Experimental Results for the comparison of EASE and SCGA

Benchmark	Measurement	SCGA	EASE	UN
F1	Mean of D	1.09E-03	**2.00E-05**	2.53E-01
	StDev of D	1.16E-03	**1.41E-05**	1.88E-02
	Mean of Peak Ratio	**1.00**	**1.00**	0.20
	StDev of Peak Ratio	**0.00**	**0.00**	0.00
F2	Mean of D	2.59E-01	**2.21E-12**	4.86E+00
	StDev of D	1.17E-01	**1.49E-11**	3.17E-01
	Mean of Peak Ratio	0.44	**1.00**	0.25
	StDev of Peak Ratio	0.18	**0.00**	0.00
F3	Mean of D	2.03E-02	**2.25E-05**	7.18E-01
	StDev of D	2.22E-02	**1.13E-04**	2.09E-06
	Mean of Peak Ratio	0.95	**1.00**	0.50
	StDev of Peak Ratio	0.15	**0.00**	0.00
F4	Mean of D	6.63E-01	**9.78E-06**	6.90E+00
	StDev of D	5.91E-01	**4.91E-05**	1.33E+00
	Mean of Peak Ratio	0.41	**1.00**	0.33
	StDev of Peak Ratio	0.14	**0.00**	0.00
F5	Mean of D	8.76E-03	**1.15E-14**	1.71E-02
	StDev of D	1.11E-02	**3.00E-14**	1.65E-02
	Mean of Peak Ratio	1.00	**1.00**	1.00
	StDev of Peak Ratio	0.00	**0.00**	0.00
F6	Mean of D	2.11E+00	**1.59E-06**	7.53E+01
	StDev of D	9.10E-01	**1.12E-05**	1.09E-05
	Mean of Peak Ratio	0.34	**1.00**	0.33
	StDev of Peak Ratio	0.05	**0.00**	0.00
F7	Mean of D	4.07E-01	**2.47E-07**	8.04E+00
	StDev of D	1.09E-01	**1.28E-06**	8.20E-01
	Mean of Peak Ratio	0.27	**1.00**	0.07
	StDev of Peak Ratio	0.08	**0.00**	0.00
F8	Mean of D	5.20E-01	**2.01E-05**	7.81E+00
	StDev of D	1.19E-01	**5.43E-05**	6.01E-01
	Mean of Peak Ratio	0.20	**1.00**	0.07
	StDev of Peak Ratio	0.06	**0.00**	0.00

4.4 A Crowding Genetic Algorithm with Spatial Locality for Multimodal Optimization

Nevertheless, as EASE is built on SCGA, it also inherits the defects from SCGA. The parameter, species distance, needs to be tuned well before applying it to a problem. In other words, if the fitness landscape of a problem is not well understood, then it is very likely that EASE will fail in the problem. Thus we propose another evolutionary algorithm for multimodal optimization in order to mitigate the parameter problem in this section. The algorithm is called Crowding Genetic Algorithm with Spatial Locality (CrowdingGA-L), which is built on CrowdingGA and does not require any extra parameter other than the conventional genetic parameters.

4.4.1 Background

4.4.1.1 Crowding Genetic Algorithm

To extend the capability of genetic algorithm, De Jong incorporates the crowding technique [98] into genetic algorithm (CrowdingGA) for multimodal optimization. Although an intense computation is accompanied, it can effectively transform genetic algorithm into an algorithm specialized for multimodal optimization. To determine the dissimilarity (or distance) between two individuals, the dissimilarity measurement proposed in Goldberg et al. [60] and Li et al. [99] is adopted. The dissimilarity between two individuals is based on their Euclidean distance. The smaller the distance, the more similar they are and vice versa.

4.4.1.2 Locality of Reference

Locality of Reference [113] (or **The Locality Principle** [114]), is one of the most fundamental principles widely used in computing. The principle was originated from memory management methods in order to predict which memory entries would be referenced soon. The main idea is to make use of neighborhood relationships for prediction, optimizing the throughput. To define the neighborhood relationship, time and space are typically taken as the proximity measures. If time is taken, it is called **temporal locality**. If space is taken, it is called **spatial locality**.

4.4.2 Crowding Genetic Algorithm with Spatial Locality

4.4.2.1 Motivation

If we do not apply any specific technique to maintain diversity, most evolutionary algorithms will prematurely converge and get stuck in a local optimum. To cope with the problem, the algorithms for multimodal optimization are usually equipped with their own local operations for diversity maintenance. In CorwdingGA, its local operation is the crowding technique. Thinking this technique more deeply, it is to propose a restriction on the individual replacement policy such that an individual gets replaced only when a fitter offspring is generated within the same niche. Thus the choice of the offspring generation method becomes a critical performance factor. Especially, in multimodal optimization, it is intuitive to choose close parent individuals to crossover, instead of distinct parent individuals because two individuals within the same niche is more likely to generate a better individual for convergence than the opposite case. Unfortunately, the offspring generations in CrowdingGA mainly relies on fitness-based measurements. Some offspring useless for multimodal optimization may be generated by two distinct parents with high fitness. They are not suitable to offer enough feasible replacement schemes for all individuals. Thus we propose a new method for offspring generation, in order to increase the chances for successful replacements.

4.4.2.2 Offspring generation with spatial locality

Close individuals tend to have similar characteristics. For each generation, the population can be seen to be divided into different niches (See the population snapshot example in Figure 4.2). Within each niche, the individuals exhibit similar positions and step-sizes for improvement. After several generations, the difference between niches may be even larger. It will be a disaster if a single evolutionary strategy is applied to all of them regardless of their niches. Luckily, it is a two-edged sword. Such property also gives us spatial locality: crossovers between close individuals can have higher chances to generate better offsprings.

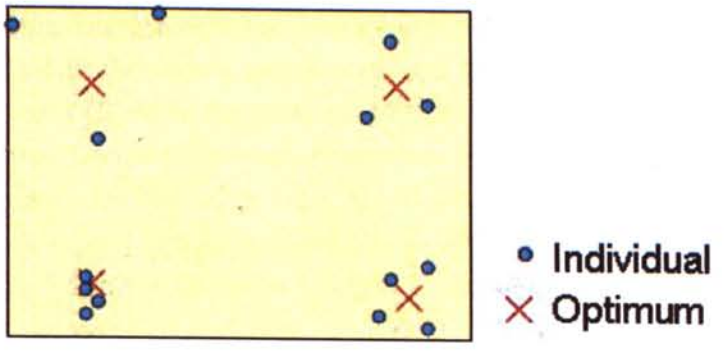


FIGURE 4.2: A Population Snapshot Example (for illustrative purpose only)

Thus a local operation is proposed to take advantage of it: the individuals which are close together should be given more chances to crossover with each other. In other words, to bring such neighborhood idea into the offspring generation, the distances between an individual and its candidate individuals are first computed. Then the distances are transformed into the proportions of a roulette-wheel [57]. Within the roulette-wheel, larger proportions of the roulette-wheel are given to closer candidate individuals. It follows that closer individuals are given higher chances for offspring generations. For the sake of clarity, the local operation is outlined in Algorithm 10.

Combined with the local operation, Crowding Genetic Algorthtm (CrowdingGA) is reformulated as a hybrid algorithm which takes advantages of spatial locality. Thus it is named **Crowding Genetic Algorithm using Spatial Locality (CrowdingGA-L)**. Offspring generations can be tailor-made for each individual. Fitter offsprings are more likely to be generated. More feasible replacement schemes can thus be provided.

Algorithm 10 Offspring Generation Using Spatial Locality

P_1: First Parent individual
P_2: Second Parent individual
O: Offspring individual

procedure NEWOFFSPRINGGENERATION(P_1)

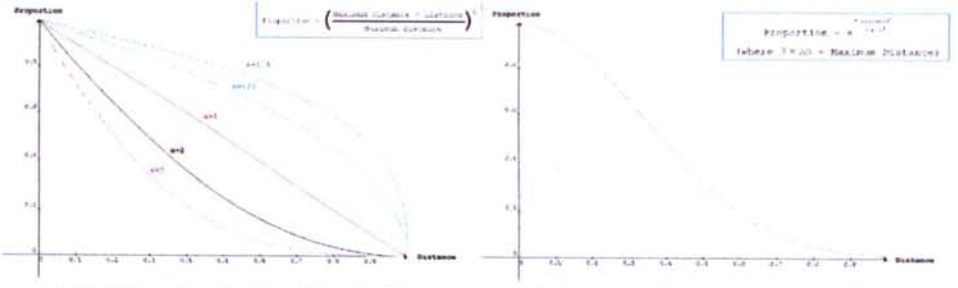
1. Transform the distances between P_1 and all candidate individuals to proportions using a transformation function;
2. Prepare a roulette-wheel based on the transformed proportions;
3. Use the roulette-wheel to pick an individuals P_2;

$O = Crossover(P_1, P_2)$;

return O;

end procedure

Mathematically, a function is needed to transform distances to proportions of a roulette-wheel. Thus two transformation functions are proposed: Since closer individuals are given higher values (proportions), the transformation function must be a monotonically decreasing function over the interval [0,MaxDistance], where MaxDistance is the maximum distance between a parent and all of its candidate individuals. Thus a simple function and Gaussian function are proposed for the transformation. The simple function is based on the formula: $Proportion = (\frac{MaxDistance - distance}{MaxDistance})^a$ where a is a scaling constant. On the other hand, the Gaussian function is based on the formula: $Proportion = exp(-(\frac{distance^2}{2 \times SD^2}))$ where $SD = \frac{MaxDistance}{3}$. Since spatial locality is normal in nature [114], the Gaussian function is adopted in CrowdingDE-L for the transformation if not specified explicitly.



(A) Five simple functions where Maximum Distance = 1

(B) Gaussian function where Maximum Distance = 1

FIGURE 4.3: Transformation Functions for Spatial Locality

4.4.3 Experiments

We implemented all the algorithms using Sun's Java programming language. The development was based on the EC4 framework [57]. Experiments to compare the performance among CrowdingGA-L and other algorithms were conducted on eight benchmark functions. The other algorithms include: Crowding Genetic Algorithm (CrowdingGA) [98], Fitness Sharing Genetic Algorithm (SharingGA) [60], SharingDE [102], Species Conserving Genetic Algorithm (SCGA) [99], and SDE [115]. The first five benchmark functions are widely adopted in literatures: F1 is Deb's 1st function [116], F2 is Himmelblau function [104], F3 is Six-hump Camel Back function [109], F4 is Branin function [109] and F5 is Rosenbrock function [110]. F6, F7, and F8 are PP1, PP3, and PP4 which were derived from [116, 117].

4.4.3.1 Performance measurements

For multimodal optimization, there are several performance metrics proposed [99, 102, 115, 117]. Our focuses are on the ability of the algorithms to locate the optima and the accuracy of the optima found by the algorithms. Hence we adopted the Peak Ratio (PR) and Average Minimum Distance to the Real Optima (D) [116, 117] as the performance metrics.

As different algorithms perform different operations in one generation, it is unfair to set the termination condition as the number of generations. Alternatively, it is also unfair to adopt CPU time, since it substantially depends on the implementation techniques for different algorithms. For instance, the sorting techniques and the programming languages used. In contrast, fitness function evaluation is always the performance bottleneck. Thus the number of fitness function evaluations was set as the termination condition in the following experiments. All algorithms were run up to a maximum of 40000 fitness function evaluations. The above performance metrics were obtained by taking the average and standard deviation of 50 runs.

4.4.3.2 Parameter setting

Sun's Java Double (double-precision 64-bit IEEE 754 floating point) was used as the real number representation for all algorithms. All populations were initialized randomly. The random seed was 12345. For all DE algorithms, the crossover probability (CR) was 0.9 and F was 0.5. The common GA parameter settings of CrowdingGA, SharingGA and SCGA for all benchmarks were the same as Table 4.2. For all crowding algorithms, population size was set to 100 for F7 and F8. 50 was set for the remaining benchmark functions. The parameter settings of SharingDE, SharingGA, SCGA, and SDE for different benchmarks are tabulated in Table 4.5. For SharingDE and SharingGA, σ and α denote the niche radius and scaling factor respectively. The parameters have been tuned in a few preliminary runs with manual inspections for all algorithms.

TABLE 4.5: Parameter setting of SharingDE, SharingGA, SCGA and SDE for different benchmarks

		SharingDE [102]		SharingGA [60]		SCGA [99] and SDE [115]
Benchmark	Population Size	α	σ	α	σ	Species Distance
F1	100	1	0.001	1	0.001	0.01
F2	100	1	0.03	5	0.1	3
F3	100	1	0.01	2	40	0.5
F4	100	1	0.01	1	0.1	6
F5	100	3	30	3	30	10
Peaks1	200	1	100	1	50	50
Peaks2	200	1	100	2	50	25
Peaks3	200	1	5	1	0.5	3
Peaks4	200	1	5	1	0.5	3
Peaks5	200	1	300	1	300	150

4.4.3.3 Results

Table 4.6 shows the experimental results for the comparison of CrowdingGA-L and the other algorithms. It can be observed that CrowdingGA-L outperformed the others in all the benchmark functions. It also improves CrowdingGA's performance no matter in the ability to locate optima or the accuracy of the optima found. Besides, UN is a canonical evolutionary algorithm in the EC4 framework. Its results can be regarded as the baseline results without any multimodal optimization techniques.

4.4.4 Conclusion

CrowdingGA-L is highlighted with its ability for effectively generating fitter offsprings. Extensive experiments have been conducted. The results indicate that CrowdingGA-L has its own competitive edge over the other algorithms tested, in terms of the performance metrics.

TABLE 4.6: Experimental Results for all algorithms tested (averaged over 50 runs)

Benchmark	Measurement	CrowdingGA-L	CrowdingGA	SharingGA	SharingDE	SDE	SCGA	UN
	Mean of D	**9.54E-07**	2.24E-06	4.08E-03	1.14E-03	1.59E-03	1.09E-03	2.53E-01
	StDev of D	3.46E-06	4.81E-06	1.21E-02	4.53E-04	7.87E-03	1.16E-03	1.88E-02
F1	Mean of Peak Ratio	**1.00**	**1.00**	0.98	**1.00**	0.99	**1.00**	0.20
	StDev of Peak Ratio	0.00	0.00	0.06	0.00	0.04	0.00	0.00
	Mean of D	**9.30E-05**	4.93E-04	2.06E+00	4.92E-01	1.20E+00	2.59E-01	4.86E+00
	StDev of D	8.28E-05	5.49E-04	1.05E+00	7.77E-01	6.36E-01	1.17E-01	3.17E-01
F2	Mean of Peak Ratio	**1.00**	**1.00**	0.66	0.91	0.78	0.44	0.25
	StDev of Peak Ratio	0.00	0.00	0.17	0.14	0.11	0.18	0.00
	Mean of D	**1.90E-05**	2.21E-05	1.44E-01	1.55E-02	6.22E-03	2.03E-02	7.18E-01
	StDev of D	2.91E-05	3.38E-05	2.89E-01	4.96E-03	2.26E-03	2.22E-02	2.09E-06
F3	Mean of Peak Ratio	**1.00**	**1.00**	0.90	**1.00**	**1.00**	0.95	0.50
	StDev of Peak Ratio	0.00	0.00	0.20	0.00	0.00	0.15	0.00
	Mean of D	**1.10E-03**	5.96E-02	3.39E+00	1.38E+00	2.61E-01	6.63E-01	6.90E+00
	StDev of D	1.25E-03	1.14E-01	1.99E+00	1.85E+00	7.81E-01	5.91E-01	1.33E+00
F4	Mean of Peak Ratio	**1.00**	0.89	0.61	0.88	0.97	0.41	0.33
	StDev of Peak Ratio	0.00	0.16	0.21	0.16	0.10	0.14	0.00
	Mean of D	**6.73E-04**	1.22E-02	8.59E-03	4.14E-02	4.23E-02	8.76E-03	1.71E-02
	StDev of D	1.36E-03	2.84E-02	1.83E-02	1.40E-01	3.07E-02	1.11E-02	1.65E-02
F5	Mean of Peak Ratio	**1.00**	0.96	**1.00**	0.92	0.94	**1.00**	1.00
	StDev of Peak Ratio	0.00	0.20	0.00	0.27	0.24	0.00	0.00
	Mean of D	**5.09E-01**	6.36E-01	4.84E+00	2.67E+01	5.00E+01	2.11E+00	7.53E+01
	StDev of D	1.61E+00	1.18E+00	7.78E+00	3.12E+01	7.71E+00	9.10E-01	1.09E-05
F6	Mean of Peak Ratio	**0.93**	0.86	0.01	0.37	0.21	0.34	0.33
	StDev of Peak Ratio	0.13	0.17	0.05	0.10	0.16	0.05	0.00
	Mean of D	**3.47E-02**	9.15E-02	1.59E+00	2.24E-01	3.69E+00	4.07E-01	8.04E+00
	StDev of D	5.20E-02	8.21E-02	3.66E-01	1.28E-01	6.48E-01	1.09E-01	8.20E-01
F7	Mean of Peak Ratio	**0.95**	0.84	0.61	0.55	0.34	0.27	0.07
	StDev of Peak Ratio	0.05	0.09	0.07	0.12	0.05	0.08	0.00
	Mean of D	**2.04E-01**	2.69E-01	2.49E+00	4.90E-01	3.36E+00	5.20E-01	7.81E+00
	StDev of D	1.69E-01	1.75E-01	4.32E-01	1.81E-01	6.96E-01	1.19E-01	6.01E-01
F8	Mean of Peak Ratio	**0.87**	0.69	0.24	0.33	0.33	0.20	0.07
	StDev of Peak Ratio	0.06	0.08	0.09	0.12	0.04	0.06	0.00

The locality principle is proven simple and useful in computing [114]. In a macro-view, the work in this section can be regarded as a case study for integrating the locality principle into an evolutionary algorithm. The numerical results can also be viewed as a valuable resource for comparing the state-of-the-art algorithms for multimodal optimization.

Chapter 5

Generalizing Protein-DNA Binding Sequence Representations and Learning using an Evolutionary Algorithm for Multimodal Optimization

In Chapter 3, we have described a set of discovered TF-TFBS sequence pairs which have been verified by PDB. In Chapter 4, we have described two evolutionary algorithms designed for multimodal optimization. In particular, Crowding Genetic Algorithm with Spatial Locality (CrowdingGA-L) is found to be effective and easy to be used, in terms of the parameter settings. In this chapter, we describe how CrowdingGA-L is applied and customized as CrowdingGP-L to generalize the sequence pairs discovered such that more information and insights can be provided for biochemists.

5.1 Introduction and Background

In Chapter 3, we have described several sequence pairs discovered by a data mining framework. In the verification process, it is found that the sequence pairs reveal some true binding core contacts between TFs and TFBSs, which can drive for further knowledge in deciphering the TF-TFBS binding.

Nevertheless, the sequence pairs discovered are in one-to-one format. One TF amino acid sequence is coupled with one TFBS DNA sequence. In the biological world, a TF may bind to their promoter using several contact surface subsequences. Some surfaces of the TF may also be interacting surfaces to recruit another TF as a performing complex

[118]. For instance, McGuire et al [119] found that there were two conserved parts for the ArcA-P recognition motif in E.coli. Kato et al. proposed a novel method to identify combinatorial regulation of transcription factors and binding motifs using chromatin immunoprecipitation (ChIP) data with microarray expression data. A case study in the evolution of combinatorial gene regulation in Fungi has also been carried out by Tuch et al. [120]. Many experimental evidences can also be found in TransCompel [121] which is a comprehensive database on the composite interactions between TFs binding to their TFBSs, experimentally proved in literature. Thus several TF amino acid sequences may be coupled with several TFBS DNA sequence, instead of just one-to-one mapping. Considering the available resources and huge search space, we propose applying an evolutionary algorithm to learn generalized representations of sequence pairs. In particular, the pairs are evolved to pairs of boolean expressions of kmers by CrowdingGP-L.

Since tree structures (boolean expressions of kmers) are described in this chapter, some related reviews are conducted:

Evolving trees by evolutionary algorithms are well studied in the genetic programming [2] field. Many design issues have also been reviewed. In particular, some researchers are especially concerned about the roles of crossover and mutation. Some of them argue that crossovers are not beneficial to the evolution, whereas the others hold the opposite view [122]. Some of them also argue that mutations are not needed, whereas the others hold the opposite view [123]. Even Sean et al. and White et al. have done an extensive experiments on comparing crossovers and mutations on a series of well known problems. They can only conclude that the benefit of crossovers is problem-dependent [124, 125]. The debate is still continuing. Thus, as a compromising solution, both crossover and mutation operators are adopted in this chapter. Another important topic in genetic programming is to control the "bloat" property. In a run of genetic programming, it is often found that some unnecessary components (called "introns") are formed. It is trivial for us to think that they are not necessary, and thus not good. However some researchers suggest that the presence of the introns can deviate the crossover operators, protecting the good components [2]. Terence et al. suggested using a fitness function which penalizes trees with many introns. Rosca also suggested providing parsimony pressure on selecting trees were beneficial to grow toward the optimal structures. Nevertheless, the bloat property is a double-edged sword. Hilimi et al. made use of the bloat property to evolve some buffer overflow attack codes which can successfully hide themselves from intrusion detectors.

With a large research community studying and analysing genetic programming, genetic programming has proved successes in many applications, for instance, data mining [126], image enhancement [127], technical trading rule finding [128], and 3-D Character Animation [129].

5.2 Problem Definition

In this problem, the 131 TFBS5mer-TF5mer pairs mentioned in Chapter 3 are transformed from one-to-one mappings into many-to-many mappings, which are the pairs of boolean expressions of 5mers with the (locally) highest lift.

Definition 5.1 A boolean expression of kmers is a tree in which its parent nodes are boolean operators AND/OR and its leaf nodes are kmers

Definition 5.2 A pair of boolean expressions of kmers has two boolean expressions of kmers. One is only composed of amino acid kmers and evaluated in the amino acid sequences of transcription factors, the other is only composed of DNA kmers and evaluated in the DNA sequences of transcription factor binding sites.

5.3 Crowding Genetic Algorithm with Spatial Locality

To apply CrowdingGA-L and customized as CrowdingGP-L to learn the generalized pairs, several customizations are needed. Pairs are represented as pairs of boolean expressions of kmers in two tress so that some hierarchical information can be stored and exchanged during the evolution process. Several crossover operators and mutation operators are also proposed to evolve the pairs.

5.3.1 Representation

For each pair, we have two parts - amino acid sequence and DNA sequence. Thus we should also have such two parts in our individuals. Each individual is a pair of boolean expressions of kmers, which is modeled as two trees - an amino acid tree in TFs and a DNA tree in TFBSs. The functional nodes are some logical operators (AND, OR), whereas the terminal nodes are the kmers found in the pairs discovered in the previous chapter. Figure 5.1 shows an example. The example represents a boolean expression which the amino acid sequences RKQAE and (CEGCK or NRESA) are found in one or more TF sequences, whereas the respective DNA sequences AAACA and (CCACG or GATAA) are commonly found to co-exist with the TF tree in the respective TFBS sequences. If a TF sequence has the kmers RKQAE and CEGCK and the TF's TFBS sequences have the kmers AAACA and CCACG commonly found, then the TF is considered true for the boolean expression.

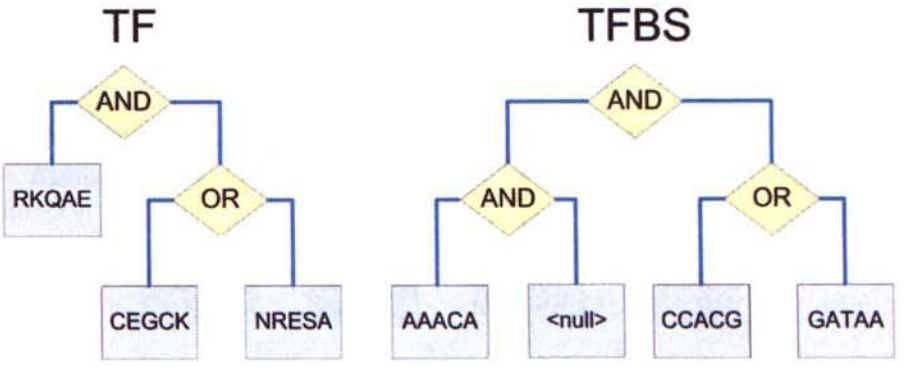


FIGURE 5.1: A individual example - a pair of boolean expressions of kmers

5.3.2 Crossover Operators

Similar to genetic programming, we adopt subtree crossover as the crossover operators. Given two trees, we randomly pick a node in each tree. Then we swap the picked nodes and its children between the two trees. Figure 5.2 depicts an example. In this example, we randomly pick the terminal node RKQAE in the left tree whereas we pick the functional node AND in the right tree as the crossover points. Then, based on the two nodes, we swap the respective subtrees rooted at the crossover points. Two new offspring trees are thus formed. But, of course, this example only shows the crossover between the trees on the TF side. It is trivial to be applied on the trees on the TFBS side. Such subtree crossover may thus be applied to the two types of trees once during each crossover operation in the evolution process.

5.3.3 Mutation Operators

Besides the crossover operators, mutation operators also play a key role in an evolution process. They can help us balance the convergence power of crossovers such that premature convergence can be avoided. Thus we propose several mutation operators in this section.

Basically, like the crossover operator, we also need to pick a random node to initiate a mutation on a tree. Based on the type of the node picked (functional node or terminal node), given the corresponding set of mutation operators, we probabilistically apply one of them on the node.

Figure 5.3 depicts the set of mutation operators on functional nodes. In particular, it shows an example on the functional node OR. If we apply insertion, then a random functional node (OR node in the example) will be inserted as the parent of the functional node. A random terminal node (RKQAE in the example) will also be inserted as a child of the functional node inserted. If we apply change, the functional node will be randomly changed to a random functional node (AND node in the example) in the same place. If

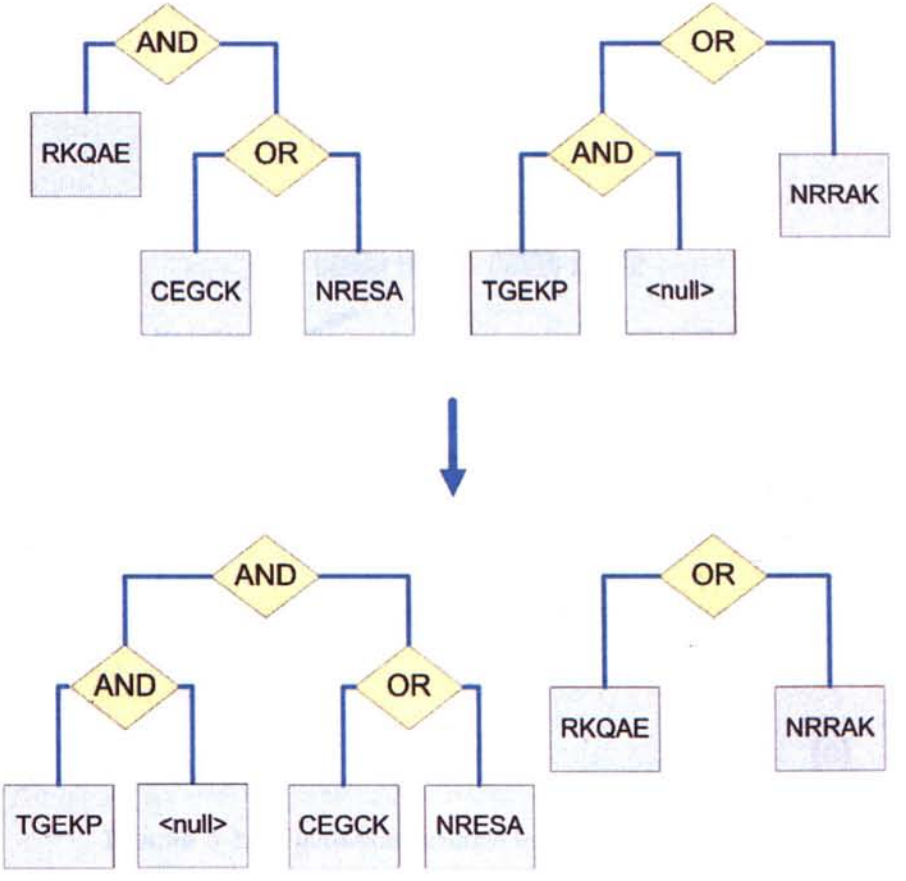


FIGURE 5.2: A crossover example on the TF side - subtree crossover

we apply deletion, the functional node and one of its child will be deleted as shown in 5.3(c).

Figure 5.4 depicts the set of mutation operators on terminal nodes. In particular, it shows an example on the terminal node CEGCK. If we apply insertion, then a random functional node (AND node in the example) will be inserted as the parent of the terminal node. A random terminal node (NRRAK in the example) will also be inserted as a child of the functional node inserted. If we apply change, the terminal node will be randomly changed to a random terminal node (NRRAK in the example) in the same place. If we apply deletion, the terminal node will be deleted as shown in 5.4(c).

5.3.4 Fitness Function

Fitness function (or objective function) is one of the most important designs in an evolutionary algorithm. It not only provides a landscape for an evolutionary algorithm, but also the direction in selecting individuals. Every individual is evaluated based on the fitness function designed.

Considering the current problem, it is trivial to adopt support as the fitness function. Nevertheless, it will encourage the evolution process to have more null terminal nodes

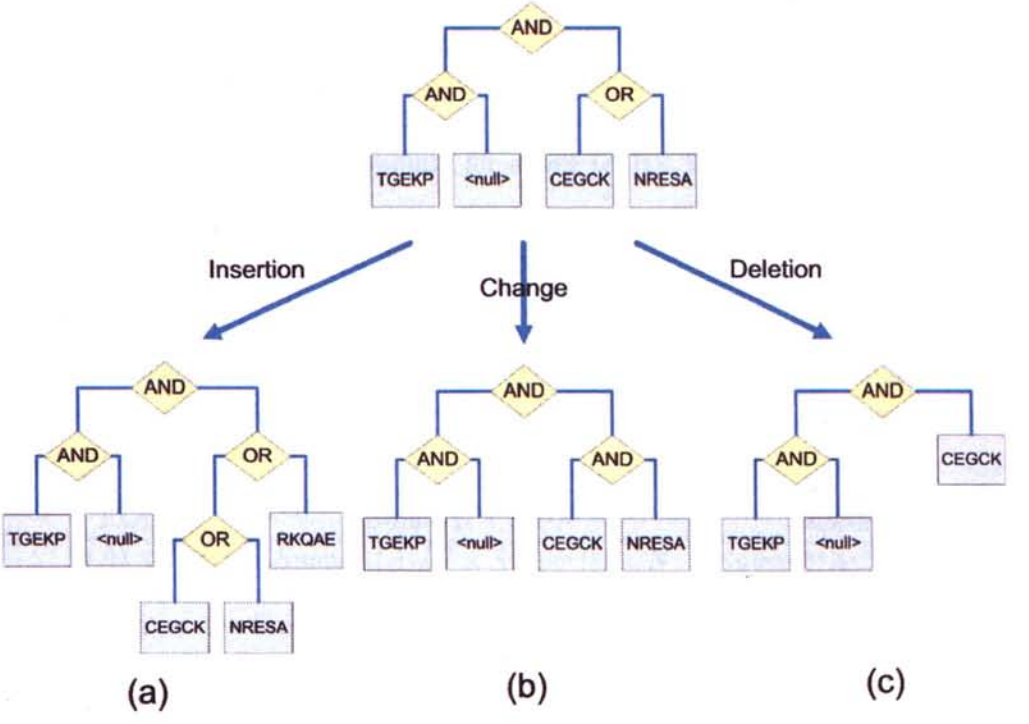


FIGURE 5.3: A mutation example in the function node OR

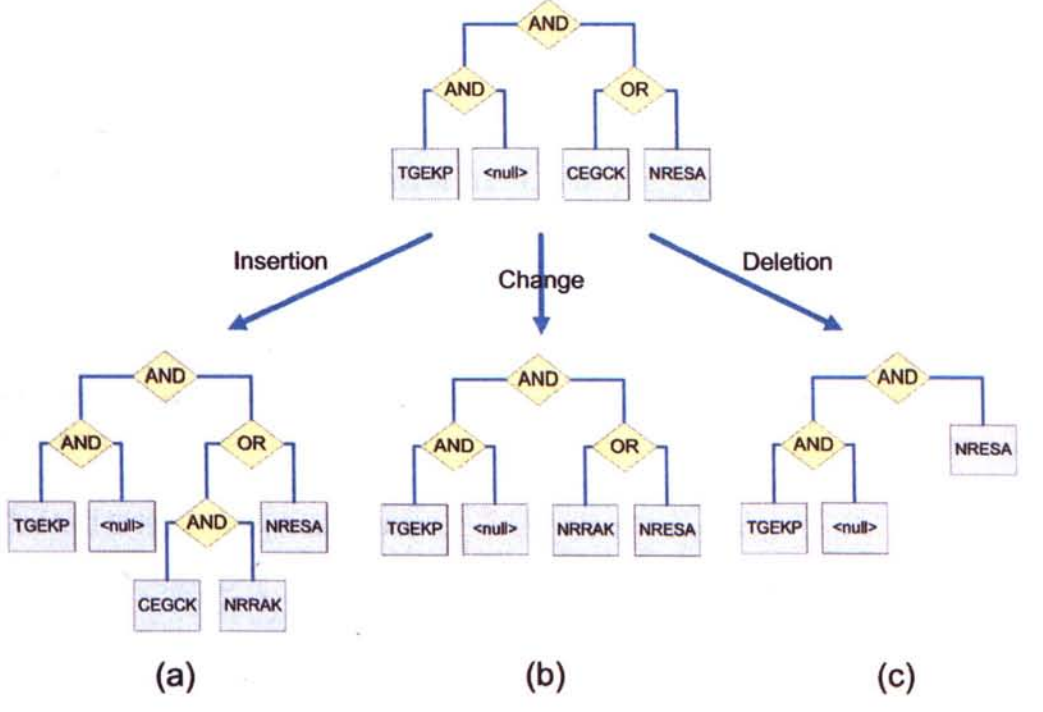


FIGURE 5.4: A mutation example in the terminal node CEGCK

and OR functional nodes. All individuals will just strive for more and more obscure condition to satisfy all the sequence data to maximize its support. Thus, obscure pairs will be generated, providing useless information for us. In contrast, lift [86] does not suffer from the issue because it estimates the ratio of the actual occurrence probability to the expected occurrence probability for a pair (A-B). The support is normalized by the background frequency:

$$lift = \frac{P(A,B)}{P(A) \times P(B)}$$

where P(A,B) is the occurrence probability of A and B, P(A) is the occurrence probability of A, and P(B) is the occurrence probability of B. Assume the total number of data samples is N. Then $P(A,B) = Support(A,B)/N$, $P(A) = Support(A)/N$, and $P(B) = Support(B)/N$. Thus

$$lift = \frac{P(A,B)}{P(A) \times P(B)} = \frac{Support(A,B)/N}{Support(A)/N \times Support(B)/N} = \frac{Support(A,B)}{Support(A) \times Support(B)} \times N$$

Thus we use it as the fitness function:

$$Fitness\ Function = \frac{Support(A,B)}{Support(A) \times Support(B)} \times N$$

To be comparable to the previous pairs generated, lift is measured only when the support is larger than or equal to seven. Otherwise, the corresponding individual will be discarded.

5.3.5 Distance Metric

In evolutionary algorithm, a distance metric can be classified as genotype, phenotype or functional distance. As their names stand, genotype, phenotype, and functional distances measure the distance between individuals in genotype, phenotype and functional spaces respectively. In this work, functional distance is adopted. The distance between two individuals is measured based on their functions, the TF sequences and the corresponding TFBS sequences where they can satisfy. The cardinality of the symmetric difference of them for each side is calculated. The value for the TF side and that for the TFBS side are combined into the formula of Euclidean distance. Square root is removed for computational efficiency. Mathematically, the distance between individual A and individual B is calculated as follows:

$$diff_{TF}(A,B) = \frac{|TF_A \ominus TF_B|}{|TF_A| + |TF_B|}$$

$$diff_{TFBS}(A,B) = \frac{|TFBS_A \ominus TFBS_B|}{|TFBS_A| + |TFBS_B|}$$

$$distance = (diff_{TF}(A,B))^2 + (diff_{TFBS}(A,B))^2$$

TABLE 5.1: EC parameter setting

Parameter	Setting
Functional Nodes	AND, OR
Terminal Nodes	kmers in Table 3.3
Population Initialization	TFBS5mer-TF5mer pairs in Table 3.3
Population Size	262
Generation Type	Overlapping [57]
Parent Selection	Binary Tournament [57]
Survival Selection	Truncation [57]
Representation	Two Trees of kmers
Mutation Type	Mixed use of the mutations mentioned
Mutation Probability	0.4
Crossover Type	Subtree Crossover [2]
Crossover Probability	0.2
Random Seed	5616516
Implementation	EC4 framework (Sun's Java programming language)

TABLE 5.2: Mutation Probability

Functional Node		Terminal Node	
Insertion	0.6	Insertion	0.6
Change	0.2	Change	0.2
Deletion	0.2	Deletion	0.2

where TF_X is the set of the TF sequences which individual X satisfies and $TFBS_X$ is the set of the TF sequences which individual X satisfies their TFBS sequences as commonly found.

5.4 Experiments

5.4.1 Parameter Setting

Table 5.1 shows the EC parameter setting for the experiments. In particular, motivated by Kraft et al., a mixed use of mutations is adopted. Given an individual, we have two trees. One tree is on the TF side, whereas the other tree is on the TFBS side. For each tree, it has a fixed and pre-defined mutation probability (0.4 in this chapter) to be mutated. If mutated, the algorithm will randomly selects a node in the tree. Based on the type of the node (functional or temrminal), the node will probabilistically undergo one of the mutations mentioned in Figures 5.3 and 5.4. These probabilities are pre-defined in Table 5.2. To control the bloat property, individuals having a tree with more than eight terminal nodes will be discarded.

5.4.2 Search Space Estimation

Totally, we have 78 terminal nodes on the TF side [< *null* >, HNLSL, IRHNL, KPPYS, NLSLN, NSIRH, PPYSY, PYSYI, QNSIR, RHNLS, SIRHN, WQNSI, PMNAF, RPMNA, CKGFF, FQNRR, NRRAK, QNRRA, QVWFQ, VWFQN, WFQNR, ARRSR, ERELK, ESARR, KQSNR, LRKQA, NRESA, QSNRE, RESAR, RKQAE, RKQSN, RLRKQ, RRSRL, RSRLR, SARRS, SNRES, SRLRK, CEGCK, CGDKA, CQYCR, CVVCG, EGCKG, FFRRT, FRRTI, GCKGF, GFFKR, GFFRR, KGFFK, KGFFR, RNRCQ, TCEGC, VCGDK, VVCGD, IWFQN, KIWFQ, NRRMK, QNRRM, GEKPY, HTGEK, TGEKP, LRYYY, LWQFL, CNACG, LCNAC, NACGL, DLVLD, FFKRS, GYHYG, ITCEG, NRCQY, RCQYC, RNQCQ, SCEGC, SGYHY, KYGQK, RKYGQ, WRKYG, NWFIN], 17 terminal nodes on the TFBS side [< *null* >, AAACA, AACAA, AAGGT, AATTA, ACGTG, AGGTC, ATTAA, CACCC, CCACG, CGGAA, CTTCC, GATAA, GCCAC, GGTCA, GTCAA, TGACA], and 2 functional nodes [AND, OR].

Let the notation $N(h_{TF}, h_{TFBS})$ be the number of possible combinations of individuals with a full tree with height h_{TF} on the TF side and with a full tree with height h_{TFBS} on the TFBS side, which is calculated as follows:

$$Assume\ h_{TF} \geq 1\ and\ h_{TFBS} \geq 1.$$

$$N(h_{TF}, h_{TFBS}) = 78^{2^{h_{TF}}} \times 2^{2^{h_{TF}-1}} \times 17^{2^{h_{TFBS}}} \times 2^{2^{h_{TFBS}-1}}$$

Based on this equation, a rough estimation of the number of possible combinations with different values of h_{TF} and h_{TFBS} is shown in Table 5.3. We can observe that the search space is quite large even when h_{TF} and h_{TFBS} are small, which motivates us to apply an evolutionary algorithm to search the space.

TABLE 5.3: Number of possible combinations of individuals with different heights

h_{TF}	h_{TFBS}	$N(h_{TF}, h_{TFBS})$	h_{TF}	h_{TFBS}	$N(h_{TF}, h_{TFBS})$
1	1	7.03E+06	3	1	1.267E+19
1	2	4.065E+09	3	2	7.324E+21
1	3	1.358E+15	3	3	2.447E+27
1	4	1.516E+26	3	4	2.731E+38
2	1	8.558E+10	4	1	2.778E+35
2	2	4.946E+13	4	2	1.605E+38
2	3	1.653E+19	4	3	5.364E+43
2	4	1.844E+30	4	4	5.987E+54

5.4.3 Experimental Procedure

With the parameter setting, we ran CrowdingGP-L on the TRANSFAC dataset described in Chapter 3 up to a maximum of one million fitness function evaluations for 50

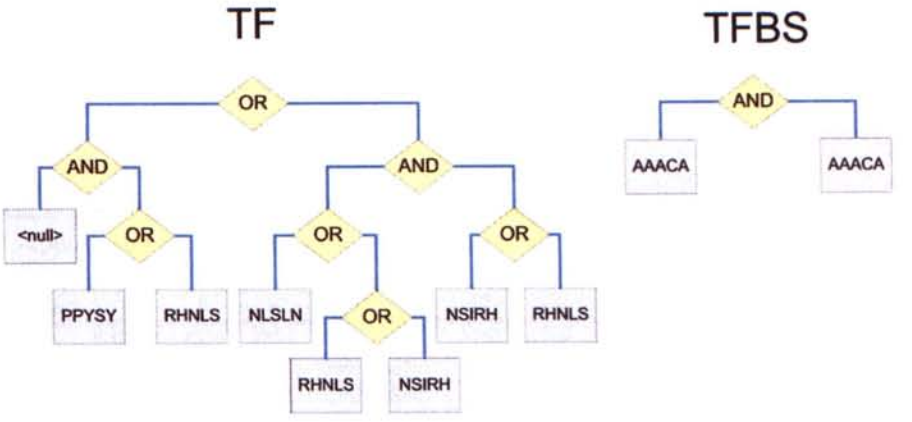


FIGURE 5.5: Generalized Example 1

runs. After each run, we picked 131 fittest pairs (the same number as the pairs in Table 3.3) out of the 262 pairs in the final population. Thus, in total, we had 6550 pairs after the 50 runs. The distribution of the pairs are tabulated in Table 5.4. For instance, 351 pairs with support$\geq$15 and lift$\geq$25 are found. Some pairs are also shown in Figure A.1.

5.4.4 Results and Analysis

5.4.4.1 Generalization Analysis

Our goal is to generalize the pairs obtained in Chapter 3. This section provides empirical analysis on three generalized examples obtained in this experiment.

The first example is depicted in Fig. 5.5. Clearly, there is only one 5mer, AAACA, on the TFBS side. This pair is, in effect, a generalized one-to-many mapping from the TFBS side to the TF side. The 5mer AAACA on the TFBS side binds to several 5mers on the TF side.

To analyze its generalization ability, we have searched through the original pairs and found that four of them were involved in this generalized pair. The four original pairs are AAACA-PPYSY, AAACA-RHNLS, AAACA-NLSLN, and AAACA-NSIRH. Each of them and the generalized pair were searched and recorded for their occurrences in the dataset as shown in Fig. 5.6.

There are four blocks divided by stars ($*$). The first block (top) denotes to which pair each column belongs (except the first column which denotes the generalized example). The second block (second from the top) provides the TF IDs which each pair (X) can satisfy on both TFBS and TF sides ($TF_X \cap TFBS_X$). The third block (third from the top) provides the TF IDs which each pair (X) can satisfy only on the TFBS side ($\neg TF_X \cap TFBS_X$). The forth block (bottom) provides the TF IDs which each pair (X) can satisfy only on the TF side ($TF_X \cap \neg TFBS_X$). The black TF IDs denote the TF

TABLE 5.4: Number of pairs learned after 50 runs across different minimal levels of support and lift. For instance, 351 pairs with support≥15 and lift≥25 are found

Lift/Support	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	6550	4555	3861	3087	2301	1781	1169	741	435	223	149	116	29	0
1	6550	4555	3861	3087	2301	1781	1169	741	435	223	149	116	29	0
2	6550	4555	3861	3087	2301	1781	1169	741	435	223	149	116	29	0
3	6550	4555	3861	3087	2301	1781	1169	741	435	223	149	116	29	0
4	6550	4555	3861	3087	2301	1781	1169	741	435	223	149	116	29	0
5	6550	4555	3861	3087	2301	1781	1169	741	435	223	149	116	29	0
6	6550	4555	3861	3087	2301	1781	1169	741	435	223	149	116	29	0
7	6550	4555	3861	3087	2301	1781	1169	741	435	223	149	116	29	0
8	6550	4555	3861	3087	2301	1781	1169	741	435	223	149	116	29	0
9	6550	4555	3861	3087	2301	1781	1169	741	435	223	149	116	29	0
10	6550	4555	3861	3087	2301	1781	1169	741	435	223	149	116	29	0
11	6550	4555	3861	3087	2301	1781	1169	741	435	223	149	116	29	0
12	6550	4555	3861	3087	2301	1781	1169	741	435	223	149	116	29	0
13	6550	4555	3861	3087	2301	1781	1169	741	435	223	149	116	29	0
14	6550	4555	3861	3087	2301	1781	1169	741	435	223	149	116	29	0
15	6550	4555	3861	3087	2301	1781	1169	741	435	223	149	116	29	0
16	6550	4555	3861	3087	2301	1781	1169	741	435	223	149	116	29	0
17	6550	4555	3861	3087	2301	1781	1169	741	435	223	149	116	29	0
18	6550	4555	3861	3087	2301	1781	1169	741	435	223	149	116	29	0
19	6550	4555	3861	3087	2301	1781	1169	741	435	223	149	116	29	0
20	6550	4555	3861	3087	2301	1781	1169	741	435	223	149	116	29	0
21	6546	4551	3857	3083	2298	1779	1168	741	435	223	149	116	29	0
22	6509	4517	3832	3064	2285	1767	1161	735	431	220	148	115	29	0
23	6433	4452	3774	3035	2264	1753	1151	729	430	220	148	115	29	0
24	6323	4343	3679	2946	2205	1706	1111	701	407	203	131	101	21	0
25	6058	4107	3459	2773	2060	1594	1035	643	***351***	147	78	61	3	0
26	5448	3667	3055	2450	1818	1371	816	427	171	26	0	0	0	0
27	3926	2183	1643	1046	588	365	180	77	22	0	0	0	0	0
28	3591	1998	1470	1003	565	352	174	73	22	0	0	0	0	0
29	2555	1535	1182	830	474	336	165	73	22	0	0	0	0	0
30	2368	1406	1090	789	437	302	156	65	15	0	0	0	0	0
31	2040	1157	872	577	386	277	132	56	15	0	0	0	0	0
32	1881	1008	752	516	328	222	122	46	5	0	0	0	0	0
33	1529	687	511	346	244	138	38	7	0	0	0	0	0	0
34	1009	515	352	201	100	65	38	7	0	0	0	0	0	0
35	884	424	261	162	63	28	1	0	0	0	0	0	0	0
36	675	224	143	74	49	28	1	0	0	0	0	0	0	0
37	533	145	66	34	9	1	0	0	0	0	0	0	0	0
38	494	122	47	20	9	1	0	0	0	0	0	0	0	0
39	461	89	41	14	5	0	0	0	0	0	0	0	0	0
40	379	58	16	10	5	0	0	0	0	0	0	0	0	0
41	246	54	15	9	5	0	0	0	0	0	0	0	0	0
42	241	49	12	6	2	0	0	0	0	0	0	0	0	0
43	210	43	11	5	1	0	0	0	0	0	0	0	0	0
44	182	15	4	4	1	0	0	0	0	0	0	0	0	0
45	153	11	0	0	0	0	0	0	0	0	0	0	0	0

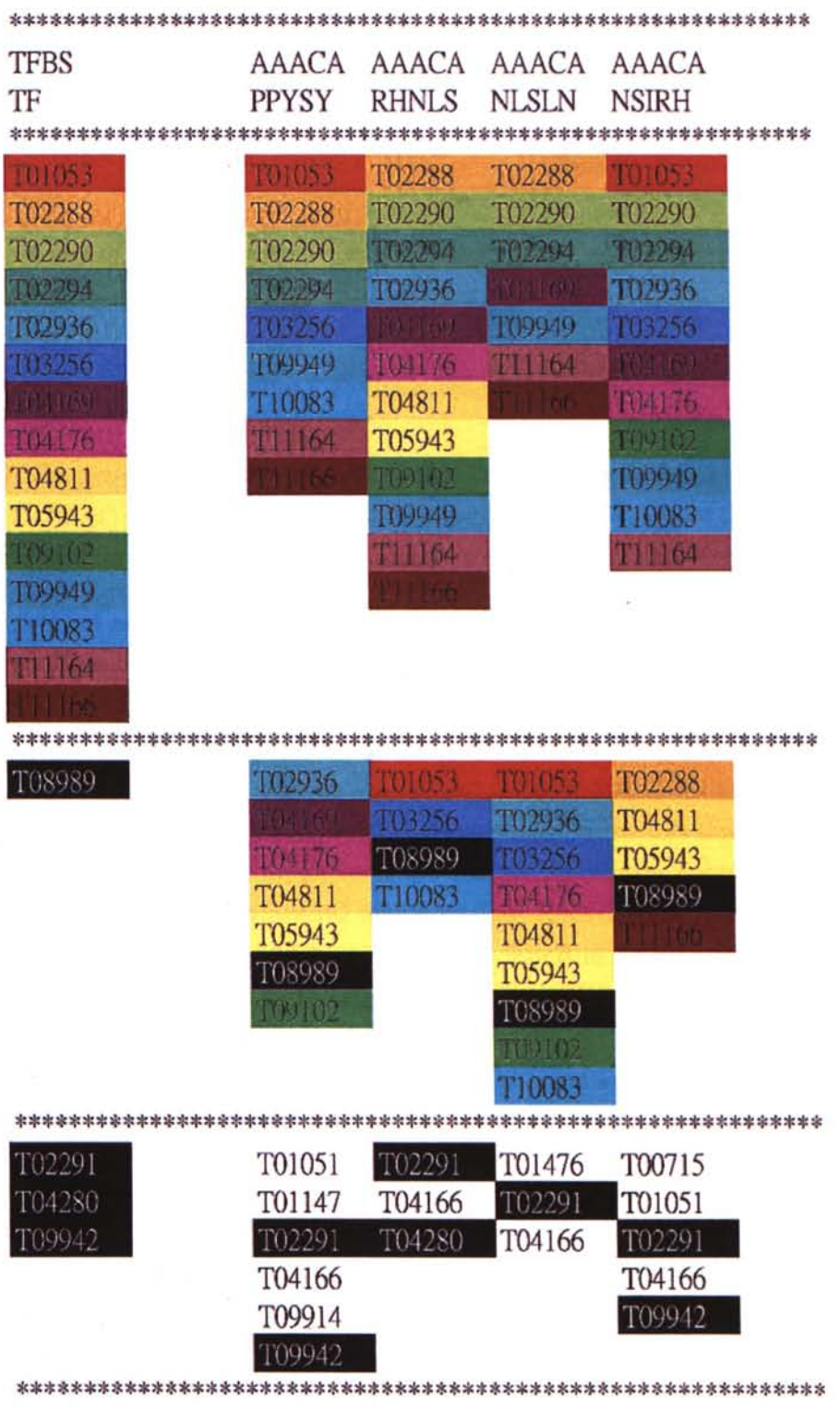


FIGURE 5.6: Generalization analysis on Generalized Example 1. There are four blocks divided by stars (∗). The first block (top) denotes to which pair each column belongs (except the first column which denotes the generalized example). The second block (second from the top) provides the TF IDs which each pair (X) can satisfy on both TFBS and TF sides ($TF_X \cap TFBS_X$). The third block (third from the top) provides the TF IDs which each pair (X) can satisfy only on the TFBS side ($\neg TF_X \cap TFBS_X$). The forth block (bottom) provides the TF IDs which each pair (X) can satisfy only on the TF side ($TF_X \cap \neg TFBS_X$). The black TF IDs denote the TF IDs which the generalized example (E) cannot fully satisfy and generalize ($(\neg TF_E \cap TFBS_E) \cup (TF_E \cap \neg TFBS_E)$), whereas the non-black TF IDs denote the TF IDs which the generalized example (E) can satisfy and generalize ($TF_X \cap TFBS_X$).

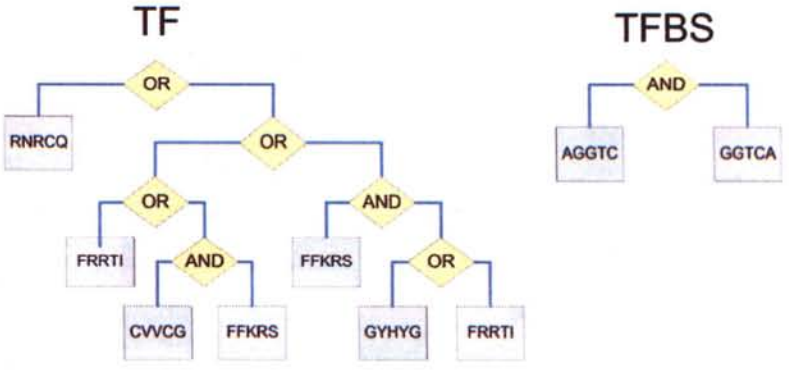


FIGURE 5.7: Generalized Example 2

IDs which the generalized example (E) cannot fully satisfy and generalize (($\neg TF_E \cap TFBS_E) \cup (TF_E \cap \neg TFBS_E)$), whereas the non-black TF IDs denote the TF IDs which the generalized example (E) can satisfy and generalize ($TF_X \cap TFBS_X$).

From Fig.5.6, we can observe that the proposed method can indeed integrate the four pairs to become a single and informative pair (Generalized Example 1). For instance, let us imagine a scenario that we only have two pairs AAACA-RHNLS and AAACA-NLSLN in mind. When we find the 5mer AAACA as commonly found in T01053 (red colour in Fig. 5.6), it is easy to deduce that we should find the two corresponding 5mers RHNLS and NLSLN in the TF sequence of T01053. However, it turns out that we cannot find any of them. By looking at Example 1, we realize the reason behind it: The 5mer AAACA can actually bind to several 5mers besides RHNLS and NLSLN. It can also bind to PPYSY and NSIRH, which can finally be found in T01053. Hence we can see that the generalized Example 1 can help us integrate and explain some datasets which the standalone one-to-one mapping pairs cannot explain. (It has also been observed that there are many movements of the non-black TF IDs not covered by the four pairs from the third block into the second block covered by the generalized pair in Fig. 5.6)

The second example is depicted in Fig.5.7. There are two similar 5mers, AGGTC and GGTCA, connected by the AND logical operator on the TFBS side. This pair is, in effect, a generalized one-to-many mapping from the TFBS side to the TF side. The 5mers AGGTC and GGTCA on the TFBS side bind to several 5mers on the TF side. Indeed, it is not hard to expect that they are most likely found as a single 6mer AGGTCA in the dataset. But, of course, it is also possible that it is not a 6mer but two very similar 5mers.

To analyze its generalization ability, we have searched through the original pairs and found that eight of them were involved in this generalized pair. The eight original pairs are AGGTC-RNRCQ, AGGTC-FRRTI, AGGTC-CVVCG, GGTCA-RNRCQ, GGTCA-FRRTI, GGTCA-CVVCG, GGTCA-FFKRS, and GGTCA-GYHYG. Each of them and the generalized pair were searched and recorded for their occurrences in the dataset as shown in Fig. 5.8.

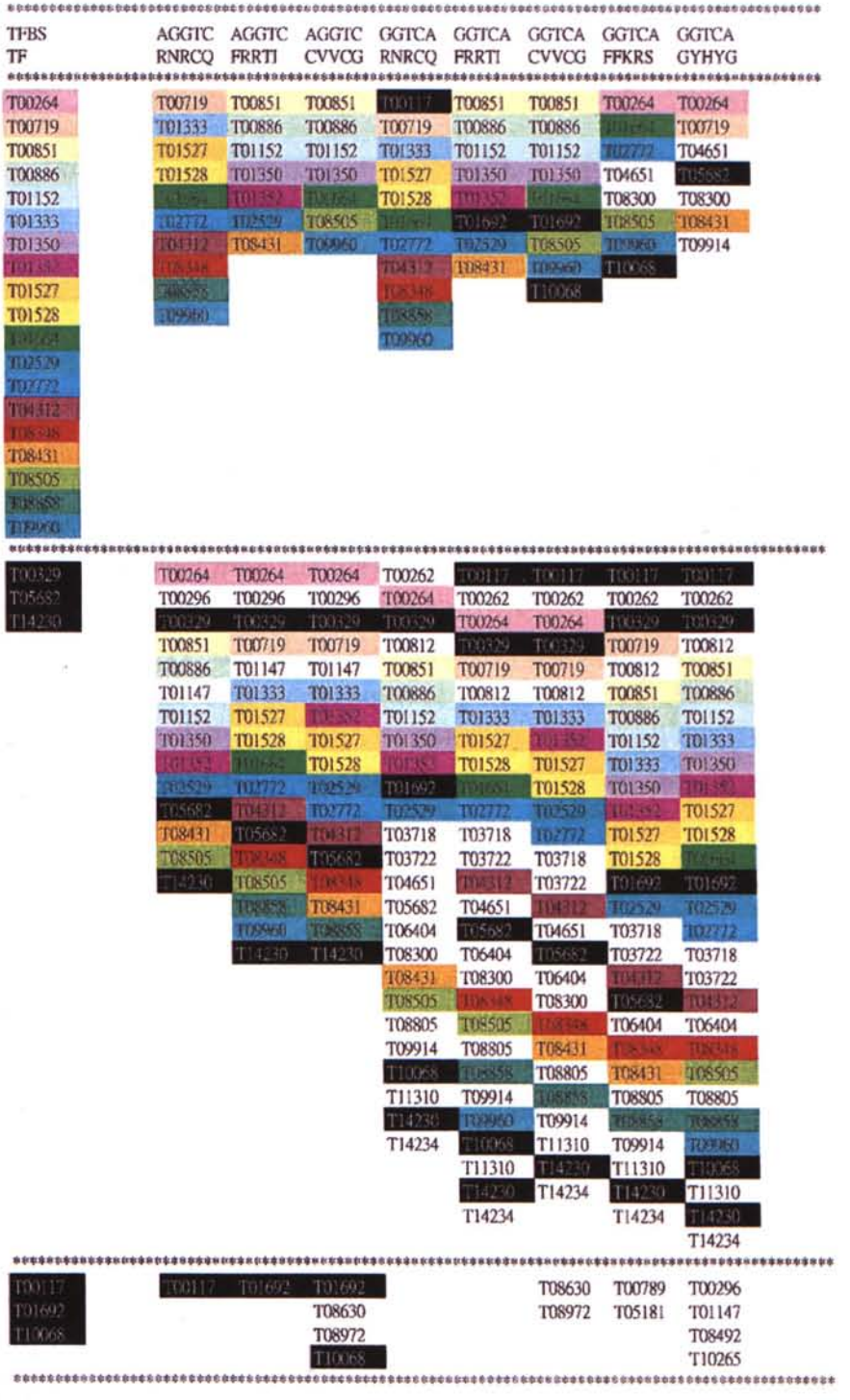


FIGURE 5.8: Generalization analysis on Generalized Example 2. There are four blocks divided by stars (∗). The first block (top) denotes to which pair each column belongs (except the first column which denotes the generalized example). The second block (second from the top) provides the TF IDs which each pair (X) can satisfy on both TFBS and TF sides ($TF_X \cap TFBS_X$). The third block (third from the top) provides the TF IDs which each pair (X) can satisfy only on the TFBS side ($\neg TF_X \cap TFBS_X$). The forth block (bottom) provides the TF IDs which each pair (X) can satisfy only on the TF side ($TF_X \cap \neg TFBS_X$). The black TF IDs denote the TF IDs which the generalized example (E) cannot fully satisfy and generalize $((\neg TF_E \cap TFBS_E) \cup (TF_E \cap \neg TFBS_E))$, whereas the non-black TF IDs denote the TF IDs which the generalized example (E) can satisfy and generalize ($TF_X \cap TFBS_X$).

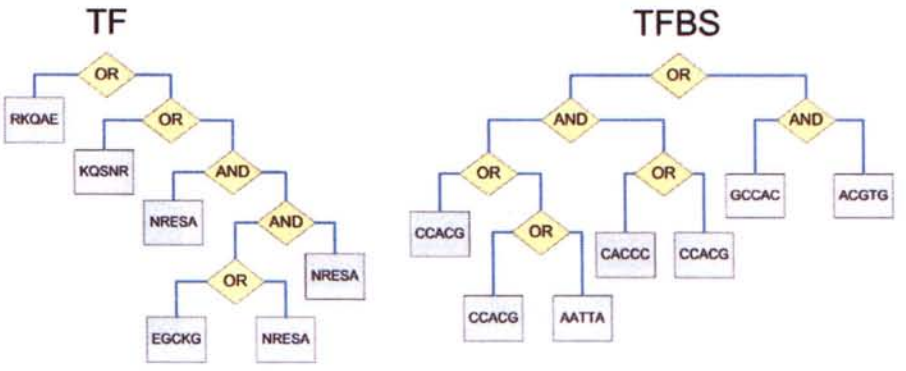


FIGURE 5.9: Generalized Example 3

There are four blocks divided by stars (∗). The first block (top) denotes to which pair each column belongs (except the first column which denotes the generalized example). The second block (second from the top) provides the TF IDs which each pair (X) can satisfy on both TFBS and TF sides ($TF_X \cap TFBS_X$). The third block (third from the top) provides the TF IDs which each pair (X) can satisfy only on the TFBS side ($\neg TF_X \cap TFBS_X$). The forth block (bottom) provides the TF IDs which each pair (X) can satisfy only on the TF side ($TF_X \cap \neg TFBS_X$). The black TF IDs denote the TF IDs which the generalized example (E) cannot fully satisfy and generalize (($\neg TF_E \cap TFBS_E) \cup (TF_E \cap \neg TFBS_E)$), whereas the non-black TF IDs denote the TF IDs which the generalized example (E) can satisfy and generalize ($TF_X \cap TFBS_X$).

From Fig.5.8, we can observe that the generalized Example 2 can indeed integrate the eight pairs to become a single and informative pair. Similar to Example 1, we can also see that the generalized Example 2 can help us integrate and explain some datasets which the standalone one-to-one mapping pairs cannot explain. (It has also been observed that there are many movements of the non-black TF IDs not covered by the eight pairs from the third block into the second block covered by the generalized pair in Fig. 5.8)

The third example is depicted in Fig.5.9. On the TF side, the sub-tree rooted at the topmost AND operator is, in effect, a single terminal node for the 5mer NRESA. Thus there are three inter-changeable 5mers, RKQAE, KQSNR, and NRESA. On the TFBS side, there are multiple combinations of 5mers. This pair is a generalized many-to-many mapping from the TFBS side to the TF side.

To analyze its generalization ability, we have searched through the original pairs and found that eight of them were involved in this generalized pair. The eight original pairs are CCACG-RKQAE, CCACG-KQSNR, CCACG-NRESA, GCCAC-KQSNR, GCCAC-NRESA, ACGTG-RKQAE, ACGTG-KQSNR and ACGTG-NRESA. Each of them and the generalized pair were searched and recorded for their occurrences in the dataset as shown in Fig. 5.8.

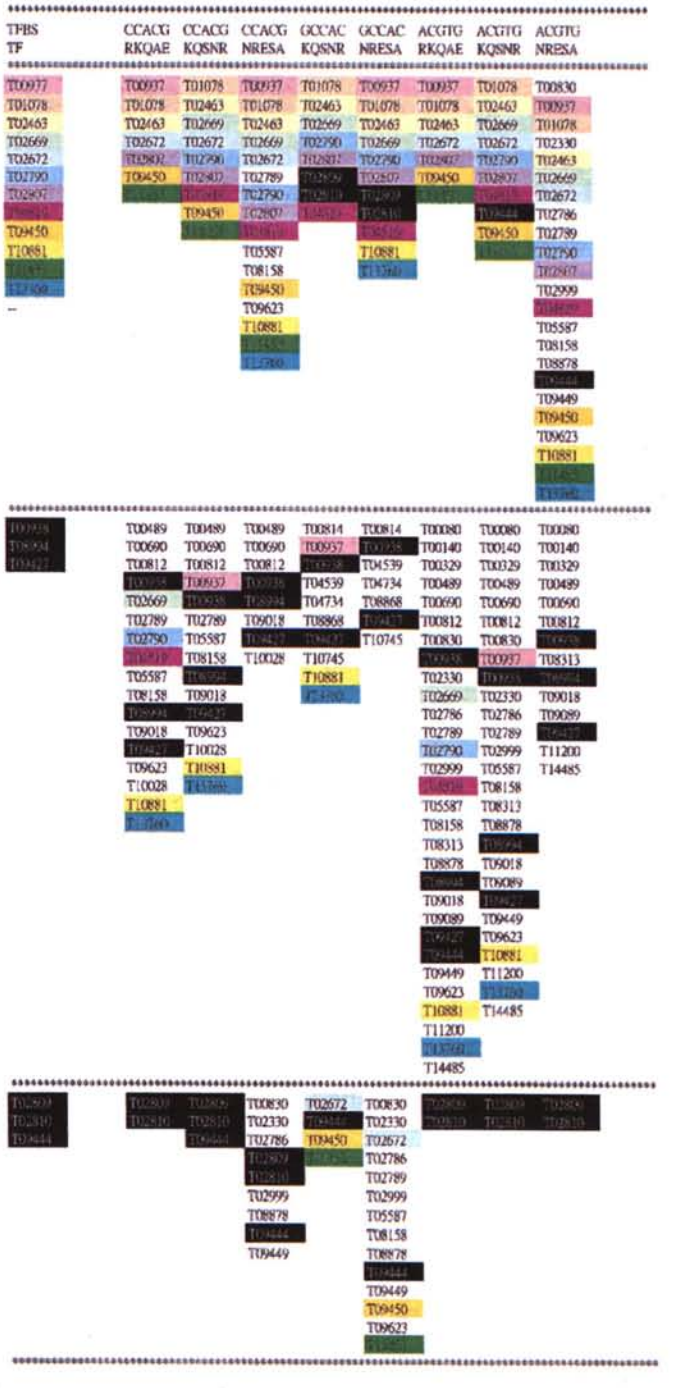


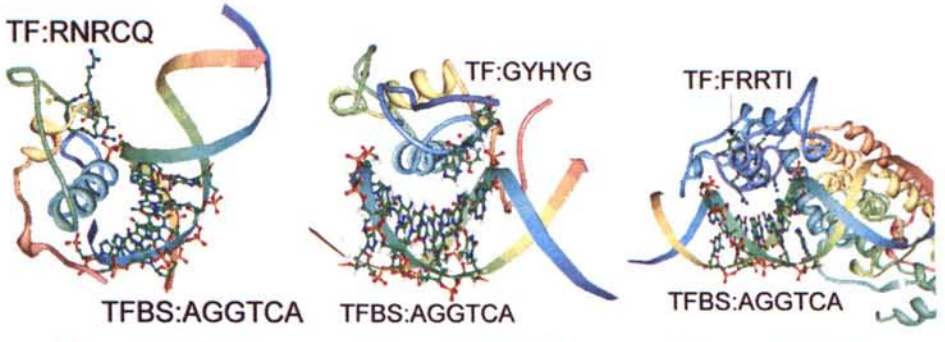
FIGURE 5.10: Generalization analysis on Generalized Example 3. There are four blocks divided by stars (∗). The first block (top) denotes to which pair each column belongs (except the first column which denotes the generalized example). The second block (second from the top) provides the TF IDs which each pair (X) can satisfy on both TFBS and TF sides ($TF_X \cap TFBS_X$). The third block (third from the top) provides the TF IDs which each pair (X) can satisfy only on the TFBS side ($\neg TF_X \cap TFBS_X$). The forth block (bottom) provides the TF IDs which each pair (X) can satisfy only on the TF side ($TF_X \cap \neg TFBS_X$). The black TF IDs denote the TF IDs which the generalized example (E) cannot fully satisfy and generalize $((\neg TF_E \cap TFBS_E) \cup (TF_E \cap \neg TFBS_E))$, whereas the non-black TF IDs denote the TF IDs which the generalized example (E) can satisfy and generalize ($TF_X \cap TFBS_X$).

There are four blocks divided by stars (∗). The first block (top) denotes to which pair each column belongs (except the first column which denotes the generalized example). The second block (second from the top) provides the TF IDs which each pair (X) can satisfy on both TFBS and TF sides ($TF_X \cap TFBS_X$). The third block (third from the top) provides the TF IDs which each pair (X) can satisfy only on the TFBS side ($\neg TF_X \cap TFBS_X$). The forth block (bottom) provides the TF IDs which each pair (X) can satisfy only on the TF side ($TF_X \cap \neg TFBS_X$). The black TF IDs denote the TF IDs which the generalized example (E) cannot fully satisfy and generalize (($\neg TF_E \cap TFBS_E) \cup (TF_E \cap \neg TFBS_E)$), whereas the non-black TF IDs denote the TF IDs which the generalized example (E) can satisfy and generalize ($TF_X \cap TFBS_X$).

From Fig.5.10, we can observe that the generalized Example 3 can indeed integrate the eight pairs to become a single and informative pair. Similar to Example 1, we can also see that the generalized Example 3 can help us integrate and explain some datasets which the standalone one-to-one mapping pairs cannot explain. (It has also been observed that there are many movements of the non-black TF IDs not covered by the eight pairs from the third and forth block into the second block covered by the generalized pair in Fig. 5.10)

5.4.4.2 Verification By PDB

Example 2 has been picked up for in-depth 3D binding analysis. Figure 5.11 depicts three respective TF-TFBS pairs, AGGTCA-RNRCQ, AGGTCA-FRRTI, and AGGTCA-GYHYG, constituting Example 2. From these figures, it can be observed that there are binding evidences for Example 2. In particular, the 6mer AGGTCA can bind to three different 5mers RNRCQ, FRRTI, and GYHYG in different situations.



(A) AGGTCA-RNRCQ pair in 1BY4

(B) AGGTCA-GYHYG pair in 1LO1

(C) AGGTCA-FRRTI pair in 3E00

FIGURE 5.11: Three representative TF-TFBS pairs for Example 2 are shown in ribbon diagram. The TF amino acids and TFBS nucleotides are highlighted in ball and stick format. The sequences of the TF-TFBS pairs are also labeled in the figures. The figures are generated using Protein Workshop [1]

Quantitatively, Table 5.5 shows the performance results obtained by the 6550 pairs in a comparison to the original 131 TFBS5mer-TF5mer pairs in Table 3.3. It can be observed that CrowdingGP-L improves the pairs in terms of the performance metrics. For the average TFBS Prediction Score, their difference is considered statistical significance with two-tailed P value = 0.002. For the average TFBS Binding Prediction Score, their difference is considered statistical significance with two-tailed P value = 0.0007. For the average Binding Prediction Score, their difference is not considered statistical significance with two-tailed P value = 0.3325.

TABLE 5.5: Performance comparison between the original pairs and generalized pairs

	Original	Generalized by CrowdingGP-L
Average TFBS Prediction Score	0.41±0.41	0.53±0.44
Average TFBS Binding Prediction Score	0.24±0.30	0.33±0.30
Average Binding Prediction Score	0.36±0.39	0.39±0.35
Percentage of verified pairs	81%	99%

5.5 Conclusion

In this chapter, we have described how CrowdingGP-L is applied to generalize the TFBS5mer-TF5mer pairs from one-to-one mappings into many-to-many mappings. It has demonstrated its potential in post-processing and generalizing the pairs. The performance measurements suggests that CrowdingGP-L can discover pairs with more generalized relationships verifiable in PDB.

Chapter 6

Predicting Protein Structures on a Lattice Model using an Evolutionary Algorithm for Multimodal Optimization

This chapter considers protein structure prediction as a multimodal optimization problem. In particular, de novo protein structure prediction problems on 3D Hydrophobic-Polar (HP) lattice model are tackled by Crowding Genetic Algorithm with Spatial Locality (CrowdingGA-L) which has been designed and applied in the previous chapters. The experimental results indicate that the proposed algorithm can be applied and shown comparable results with the state-of-the-arts algorithms in terms of the performance metrics used, even though it is relatively simple.

6.1 Introduction

A polypeptide is a chain of amino acid residues. Once folded into its native state, it is called protein. Proteins plays vital roles in living organisms. They perform different tasks to maintain a body's life. For instance, material transportations across cells, catalyzing metabolic reactions and body defenses against viruses. Nevertheless, functions of proteins substantially depend on their structural features. In other words, researchers need to know a protein's native structure before its function can be completely deduced. It gives rises to the protein structure prediction problem.

The protein structure prediction problem is often referred as the "holy grail" of biology. In particular, Anfinsen's dogma [130] and Levinthal's paradox [131] play important roles in this problem. Anfinsen's dogma postulates that a protein's native structure (tertiary

structure) only depends on its amino acid residue sequence (primary structure). On the other hand, Levinthal's paradox postulates that it is too time-consuming for a protein to randomly sample all the feasible confirmation regions for its native structure. But, on the other hand, the proteins in nature can still spontaneously fold into its native structures in about several milliseconds.

Based on the above ideas, researchers have explored the problem throughout several years. In particular, the designability of a structure and the degeneracy of a sequence have been studied by Li et. al. [132]. The computational complexity has also been examined by Hart et. al. [133].

Numerous prediction approaches have been proposed. In general, they can be classified into two categories, depending on whether any prior knowledge other than sequence data has been incorporated [134]. This paper focuses on De novo (or Ab initio) protein structure prediction on 3D Hydrophobic-Polar (HP) lattice model using evolutionary algorithms [58]. In other words, only sequence data is considered.

Different protein structure models have been proposed in the past [135]. Their differences mainly lies in their resolution levels and search space freedom. For the highest resolution levels, all the atoms and bond angles can be simulated using molecular dynamics. Nevertheless, there is no free lunch. The simulation is hard to be completed by the current computational power. On the other hand, a study indicated that protein folding mechanisms might be simpler than previously thought [136]. Simplified models are enough. Thus this paper focuses on HP lattice model to capture the physical principles of protein folding process [137, 138].

6.2 Problem Definition

In this problem, it assumes that the main driving forces are the interactions among the hydrophobic amino acid residues. The twenty types of amino acids are experimentally classified as either hydrophobic (H) or polar (P). An amino acid sequence is thus represented as a string $\{H, P\}^+$. Each residue is represented as a non-overlapping bead in a cubic lattice L. Each peptide bond in the main chain is represented as a connecting line. A protein is thus represented as a non-overlapping chain in L.

Based on the above model, the objective of the protein structure prediction problem is to find the conformation with the minimal energy for each protein. Mathematically, it is to minimize the following function [132]:

$$H = \sum_{i+1<j} E_{\sigma_i \sigma_j} \Delta(r_i - r_j)$$

where r_i and r_j are amino acid residues at sequence position i and j. The constraint $i+1 < j$ is to ensure that r_i and r_j are not next to each other in their sequence and they are examined together once only. $\Delta(r_i - r_j) = 1$ when r_i and r_j are adjacent in L, otherwise $\Delta(r_i - r_j) = 0$. As stated in the previous section, each residue is represented as either H or P. Thus $E_{\sigma_i\sigma_j}$ could be E_{HH}, E_{HP}, E_{PH}, or E_{PP}. For their values, three schemes have been proposed. The most widely used scheme is $E_{HH} = -1, E_{HP} = 0, E_{PH} = 0, E_{PP} = 0$. The second scheme $E_{HH} = -2.3, E_{HP} = -1, E_{PH} = -1, E_{PP} = 0$ was proposed by [132]. The last scheme $E_{HH} = -2, E_{HP} = 1, E_{PH} = 1, E_{PP} = 1$ is called functional model protein (or "shifted" HP model) [139]. As mentioned in [135], the results are insensitive to the value of E_{HH} as long as the physical constraints [132] are satisfied. Thus we have chosen the first scheme in the following sections.

6.3 Representation

For the representation of an amino acid residue sequence, there are two conditions to be satisfied: [58]

1. Sequence connectivity
2. Self-avoidance

Among the representations proposed [139], *Internal Coordinate* should be a favorable choice since it can handle the first condition implicitly. Internal coordinate is a representation system which residue positions depend on their sequence-predecessor residues. There are two types of internal coordinate representation: *Absolute Encoding* and *Relative Encoding*. Absolute encoding represents each residue position as the absolute direction from the previous residue. A sequence is represented as $\{U, D, L, R, F, B\}^{n-1}$ (Up, Down, Left, Right, Forward, Backward) [140]. On the other hand, relative encoding represents those as relatively directional changes based on the directions of the two predecessor residues. Backward direction is omitted for one-step self-avoiding. Thus a sequence is represented as $\{F, R, L, U, D\}^{n-2}$ [141]. Except the forward move, a cyclic conformation is formed if a move is repeated four times. Krasnogor et al. [58] have examined both representations on square lattices. Their results showed that relative encoding had better performance than absolute encoding on square lattices. Our preliminary results also indicates that the performance of absolute encoding degrades as a sequence gets longer on cubic lattices. Thus we have chosen relative encoding as the representation in the following sections. For this representation, different orientations can be taken. Nevertheless, few explicitly stated their representations in a pictorial way. Thus the representation we have adopted is depicted in Fig.6.1 for the sake of clarity. The most left sub-figure denotes the absolute direction axis, whereas the remaining sub-figures denotes the relative encoding representations for all the six directions in cubic

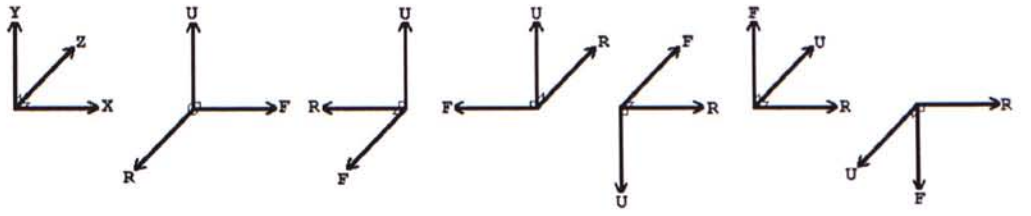


FIGURE 6.1: Relative encoding used

lattices. For instance, the second left sub-figure denotes the relative encoding representation the subsequent move should use when the current move is in the positive X direction. In particular, the subsequent move is called forward move if it is still in the positive X direction.

6.4 Related Works

Although the 3D HP model seems relatively simple among other models, it has been proved that the protein structure prediction problem on the model is NP-Complete [142]. Thus researchers propose heuristics as compromising solutions. In particular, the seminal work by Unger et al. [140] experimentally showed that genetic algorithm approaches were better than Monte Carlos simulations. Thus many researchers tried genetic algorithm as one of the heuristics to solve the problem. Nevertheless, the genetic algorithm approach by Unger et al. [140] was actually hybridized with Monte Carlo moves. Hence Patton et al. [141] further generalized it into a standard genetic algorithm approach, which search space included infeasible regions penalized by a penalty function. Furthermore, they proposed "relative encoding" so that one-step self-avoiding constraints could be implicitly incorporated in the genome representation. Few years later, Krasnogor et al. [58] published a work discussing the basic algorithmic factors affecting the problem. Since then, researchers explored different ways to tackle the problem. For instance, Krasnogor et al. further applied a multimeme algorithm, which adaptively chose multiple local searchers to reach optimal structures [143]. Cox et al. [144] and Hoque et al. [145] utilized heavy machinery of specific genetic operators and techniques. Ant colony algorithm [146], differential evolution [147], immune algorithm [139] and estimation of distribution algorithm [138] were also customized and reported in literatures. In particular, diversity preserving techniques were often incorporated in them. For instance, Duplicate Predator [144], Aging Operator [139], and additional renormalization of the pheromone [146]. They can be deemed as the signs of the multimodality in the problem. However, to the authors' knowledge, the necessity of multimodal optimization techniques has not been emphasised.

6.5 Crowding Genetic Algorithm with Spatial Locality

6.5.1 Motivation

For the protein structure prediction problem, it is generally believed that the native state of a protein should be in the conformation with the lowest energy. Thus previous works mainly focus on the minimal energy they could achieve: the minimal energy ever found ($H(x)$) and the average and standard deviation of the minimal energy across several runs ($mean \pm \sigma$).

Nevertheless, Jahn et al. [148] has shown that the native state is not necessarily a single global optimum. It may also be a local optimum in Fig.1 of [148]. For the HP lattice model, Unger et al. [149] have observed that there can be multiple conformations for each energy value. A recent fitness landscape study also indicated that HP landscapes were highly multimodal [150].

Thus we propose applying Crowding Genetic Algorithm with Spatial Locality (CrowdingGA-L) presented in Chapter 4 to the problem for multimodal optimization explicitly, in order to preserve diversity. In other words, building blocks and optima can be preserved. A more effective search is guaranteed throughout each run. Both global and local optima are more likely to be found. The native state information is less likely to be lost.

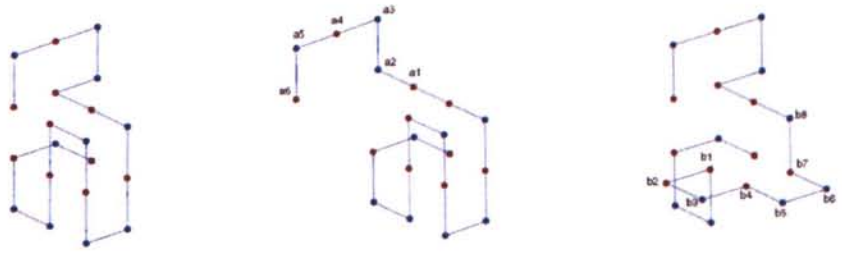
6.5.2 Customization

As CrowdingGA-L was originally designed for real number optimization, careful modifications are needed before applying them to protein structure prediction. In particular, there are two critical factors to be considered: Distance metrics and Handling infeasible conformations.

6.5.2.1 Distance metrics

The most widely used distance measure should be the root mean square deviation (RMSD) [151]. RMSD calculates the average absolute distances between two superimposed conformations' points. Nevertheless, if two conformations differ by only one point direction in relative encoding, their RMSD cannot reflect such small change. For instance, some conformations of the benchmark UM20 [152] are visualized in Fig.6.2. Fig.6.2a depicts one of the optimal conformations. The other sub-figures depict two candidate conformations:

- `Optimal : LDLDFLUFDDFRFRDDFD`
- `Example A: LDLDFLUFDDFRFFDDFD`



(A) An optimal conformation (B) Example A (C) Example B

FIGURE 6.2: Some conformations of UM20

- `Example B: LDLDDLLRLLDRFRDDFD`

To be mutated to the optimal conformation, Example A is only needed to change its move between a1 and a2 to R whereas example b is needed to change nearly all of its moves between b1 and b8. However, the RMSD of Example A with the optimal conformation (5 diagonal point changes a2 to a6) is larger than that of example B (4 diagonal point changes b2,b3,b5,b6). RMSD cannot capture the move information in relative encoding.

Furthermore, if RMSD is applied in our algorithms, it will be quite computationally intensive. To calculate the RMSD between two conformations, the corresponding relative encoding genomes are converted to absolute 3D coordinates. Once converted, one of them is then translated and rotated to be optimally superimposed on the other. RMSD is then calculated which involves multiplications and square root calculations. In contrast, Hamming distance calculates the move differences between two relative encoding genomes. It is relatively computational tractable. Thus Hamming distance is adopted as the distance metric in this chapter. [1]

6.5.2.2 Handling infeasible conformations

Basically there are two approaches:

- Delete infeasible conformations
- Tolerate infeasible conformations by adjusting their energy values by a penalty score (either constant or adaptive)

[1] Note that we have implicitly handled rotational symmetry by omitting the first move in the relative encoding representation. That's why a confirmation is represented as $\{F, R, L, U, D\}^{n-2}$, instead of $\{F, R, L, U, D\}^{n-1}$ in the previous sections. In other words, a single sequence in relative encoding representation actually represents the confirmations for all the six rotational directions in cubic lattices.

Both approaches were thought beneficial in different view angles [58, 135, 140, 150]. For the first approach, it is conjectured that search space can be smaller if infeasible conformations are deleted. For the second approach, it is conjectured that the paths to optimal conformations are shorter if infeasible conformations exist. Nevertheless, the study in [150] had a detailed analysis supporting the first approach. Furthermore, our problem is a discrete optimization problem. Unlike continuous optimization, its gene can easily flip between different values. There may be alternative paths to optimal conformations even if infeasible conformations are disallowed. Thus the first approach is taken.

Having decided the distance and infeasible confirmation handling methods, CrowdingGA-L is applied to the protein structure prediction problem on several benchmark sequences in the following sections.

6.6 Experiments

6.6.1 Performance Metrics

The widely used performance metrics are adopted [138, 139, 152]. The energy of the best conformation found ($H(x)$) indicates the best convergence an algorithm can achieve across several runs, whereas the mean and standard deviation of the minimal energy across several runs ($mean \pm \sigma$) report the stochastic convergence behavior of an algorithm.

6.6.2 Parameter Settings

The parameter settings for CrowdingGA-L in all benchmarks are tabulated in Table 6.1. CrowdingGA-L has been implemented in the EC4 framework [57] for this problem. With overlapping generation type and high selection pressure imposed in survival selection, mutation probability was set to a high value for achieving global search capability. Thus 0.8 was adopted. Crowding factor was set to population size to avoid replacement error. To be comparable to the state-of-the-art algorithms [138, 139, 152], CrowdingGA-L was run 50 times up to 10^5 and 5×10^6 energy evaluations respectively. The benchmarks were taken from [138, 139, 152].

6.6.3 Results

Table 6.2 and Table 6.3 show the experimental results which were run up to 10^5 and 5×10^6 energy evaluations respectively. For each benchmark, the performance metrics discussed have been calculated. For instance, looking at Table 6.2 and sequence s1,

TABLE 6.1: Parameter Setting for CrowdingGA-L

Parameter	Setting
Population Initialization	Straight line (FFFF....FF)
Population Size	100
Generation Type	Overlapping
Parent Selection	Uniform Deterministic
Survival Selection	Truncation/Crowding
Mutation Type	Bit Flip
Mutation Probability	0.8
Crossover Type	Two Point Crossover
Crossover Probability	1
Random Seed	123
Implementation	EC4 framework [57]

TABLE 6.2: Experimental Results of the state-of-the-art algorithms (10^5 energy evaluations)

		Hybrid GA [152]	IA [139]	MK-EDA2 [138]		TreeEDA [138]		MT-EDA4 [138]		CrowdingGA-L
s1	H(x)	**-11**	**-11**	**-11**	**-11**	**-11**	**-11**	**-11**	**-11**	**-11**
	mean±σ	-9.84±0.86	-10.90±0.32	**-11.00±0.00**	**-11.00±0.00**	**-11.00±0.00**	-10.96±0.04	**-11.00±0.00**	**-11.00±0.00**	-10.68±0.47
s2	H(x)	-11	**-13**	**-13**	**-13**	**-13**	**-13**	**-13**	**-13**	**-13**
	mean±σ	-10±0.87	-12.22±0.65	-12.94±0.09	**-13.00±0.00**	-12.86±0.16	-12.96±0.04	-12.70±0.50	**-13.00±0.00**	-11.46±0.81
s3	H(x)	**-9**	**-9**	**-9**	**-9**	**-9**	**-9**	**-9**	**-9**	**-9**
	mean±σ	-8.64±0.69	-8.88±0.48	-8.94±0.06	-8.96±0.04	-8.90±0.09	-8.98±0.02	-8.98±0.02	-8.98±0.02	**-9.00±0.00**
s4	H(x)	**-18**	**-18**	**-18**	**-18**	**-18**	-17	**-18**	**-18**	**-18**
	mean±σ	-13.72±1.41	-16.08±1.02	-15.66±1.54	-15.48±0.83	-16.31±0.51	-15.00±0.86	-16.32±0.10	-15.02±0.88	**-16.50±0.95**
s5	H(x)	**-28**	**-28**	-22	-24	-27	-24	-23	-24	**-28**
	mean±σ	-18.90±2.08	-24.82±0.71	-19.66±1.37	-20.52±1.15	-23.62±1.83	-20.68±1.65	-18.44±1.60	-20.22±2.30	**-25.44±1.36**
s6	H(x)	-22	-23	**-30**	-26	**-30**	-26	-28	-24	-27
	mean±σ	-19.06±1.46	-22.08±1.43	-26.30±2.26	-23.38±1.30	-26.00±2.82	-22.08±2.48	**-26.70±1.97**	-22.54±1.27	-23.80±1.21
s7	H(x)	-38	-41	-37	-38	-37	-38	-35	-38	**-47**
	mean±σ	-32.28±3.09	-39.02±0.50	-32.66±3.13	-33.84±2.91	-32.94±1.53	-33.10±3.11	-31.72±2.98	-32.46±3.03	**-40.98±1.96**
s8	H(x)	-36	-42	-42	-40	-44	-34	-37	-37	**-47**
	mean±σ	-30.84±2.55	-39.07±1.20	-36.66±4.02	-34.66±2.60	-34.70±6.87	-30.82±2.97	-32.24±2.47	-30.96±2.47	**-41.06±2.40**

CGA-mixed has ever achieved -11 as its minimal energy across 50 runs. On average, CGA-mixed has also achieved -10.68 as its minimal energy with standard deviation 0.47.

Interestingly, although CrowdingGA-L is a relatively simple algorithm, it can still show comparable results with the state-of-the-art algorithms when the termination condition is set to 10^5 energy evaluations and 5×10^6 energy evaluations.

Its effectiveness is largely due to their individual replacement technique: crowding. In this technique, a conformation cannot replace a dissimilar conformation. It gives freedom for all niches to evolve to their respective optima. Diversity is adaptively preserved. In particular, such diversity prevent a population from genetic drift. Useful sub-conformations (like secondary structures [134]) can be preserved, providing the algorithm a long-term sustainability for finding multiple optima at the same time.

6.7 Conclusion

In this paper, we have modeled protein structure prediction as a multimodal optimization problem. To foster its development, CrowdingGA-L has been implemented and tested. It

TABLE 6.3: Experimental Results of the state-of-the-art algorithms (5×10^6 energy evaluations)

		Hybrid GA [152]	MK-EDA2 [138]	TreeEDA [138]	MT-EDA4 [138]	CrowdingGA-L
s1	H(x)	**-11**	**-11**	**-11**	**-11**	**-11**
	mean±σ	-10.52±0.54	-10.82±0.38	-10.68±0.51	-10.84±0.37	**-11.00±0.00**
s2	H(x)	**-13**	**-13**	**-13**	**-13**	**-13**
	mean±σ	-11.28±0.90	-12.02±0.94	-11.30±0.85	-11.88±0.93	**-12.86±0.40**
s3	H(x)	**-9**	**-9**	**-9**	**-9**	**-9**
	mean±σ	-8.54±0.64	-8.96±0.19	-8.92±0.27	**-9.00±0.00**	**-9.00±0.00**
s4	H(x)	**-18**	**-18**	**-18**	**-18**	**-18**
	mean±σ	-15.76±1.05	-16.40±0.80	-16.24±0.83	-16.50±0.96	**-17.94±0.31**
s5	H(x)	-28	**-29**	**-29**	**-29**	**-29**
	mean±σ	-24.60±1.57	-27.24±0.92	-26.88±0.93	-27.06±1.08	**-28.30±0.71**
s6	H(x)	-26	-29	**-31**	-28	-28
	mean±σ	-23.02±1.48	-25.70±1.26	-25.94±1.58	-25.74±1.22	**-26.58±0.88**
s7	H(x)	**-49**	**-49**	**-49**	-48	**-49**
	mean±σ	-41.18±2.75	**-46.30±2.04**	-43.78±3.10	-42.00±6.76	-46.12±1.39
s8	H(x)	-46	**-52**	-49	-50	-50
	mean±σ	-40.40±2.50	**-46.78±2.28**	-43.72±2.43	-45.64±2.03	-46.36±1.95

is observed that CrowdingGA-L is applied and shown experimental results with the other state-of-the-art algorithms, although it is a relatively simple technique. Such interesting results may also provide some biological implications for scientists. For instance, the importance of the existences of intermediate sub-conformations could be examined in the multimodal optimization pathways provided by CrowdingGA-L.

Chapter 7

Conclusion and Future Work

7.1 Thesis Contribution

In this thesis, a framework has been proposed to discover associated TF-TFBS binding sequence patterns from TRANSFAC. To further analyze the discovered sequence patterns in the huge search space, two evolutionary algorithms have been proposed. In particular, the evolutionary algorithms are specially designed for multimodal optimization to avoid premature convergence and genetic drift. The one with less number of parameters (CrowdingGA-L) has been selected and applied to generalize the sequence representations. Some promising results have been obtained. As a further application, CrowdingGA-L has also been applied to predict protein structures on a lattice model. Some experimental results comparable with the other state-of-the-art algorithms have been obtained by applying it, although it is a relatively simple technique. In summary, the author has made the following contributions:

- Propose a novel data mining framework to discover and validate Protein-DNA Binding sequence patterns. Due to the simplicity of the framework, it can be widely applied to similar problems, such as Protein-Protein interaction.
- Propose two novel versions of evolutionary algorithms for multimodal optimization. In particular, the author extends Species-Conserving Genetic Algorithm (SCGA) and proposes Evolutionary Algorithm with Species-Specific Explosion (EASE). The author also extends Crowding Genetic Algorithm and proposes Crowding Genetic Algorithm with Spatial Locality (CrowdingGA-L).
- Apply CrowdingGA-L to generalize the pairs discovered. Its generalization ability has been empirically analyzed by some case studies. In the PDB verification process, some promising results have been obtained.
- Apply CrowdingGA-L to predict protein structures on a lattice model. Comparable results with the state-of-the-art algorithms have been obtained.

7.2 Future Work

In the future, several future works could be investigated.

The data mining framework described in Chapter 3 can be extended for mining approximate associations. Such an extension can handle the experimental and biological noises, although the inevitable computational burden needs to be carefully handled, and much more efforts are needed to distinguish real signals from the large number of false positives introduced by loosening the pattern matching and clustering. Combinatorial associations between multiple TF and TFBS k-mers will also be another challenging topic. We will also seek further real applications of the approach on experimentally verifiable TF-TFBS bindings.

For the design of the evolutionary algorithms for multimodal optimization in Chapter 4, we will focus on experiments with real world multimodal optimization problems. High dimensional problems will also be considered. Species-specific Explosion will be investigated to improve other evolutionary algorithms. Besides spatial locality, temporal locality will also be considered in different evolutionary algorithms.

Besides, we suspect that most of the evolutionary algorithms applied to bioinformatics are always stuck in local optima. People are either not aware of the issue, or too lazy to study and handle it as long as the local optima found are good enough in practice. Thus we can foresee, if we further apply some evolutionary algorithms for multimodal optimization to the bioinformatics problems other than those in Chapter 5 and 6, promising results will be probably obtained.

Appendix A

Appendix

Algorithm 11 Species Conserving Genetic Algorithm

$G(t)$: Generation at time t
X_s: A set storing species seeds

$t \leftarrow 0$;
Initialize $G(t)$;
Evaluate $G(t)$;
while not termination condition **do**
    Identify Species Seeds X_s;
    Select $G(t+1)$;
    Crossover $G(t+1)$;
    Mutate $G(t+1)$;
    Evaluate $G(t+1)^3$;
    Conserve species from X_s in $G(t+1)$;
    $t \leftarrow t+1$;
end while
Identify species seeds X_s;
Identify global optima;

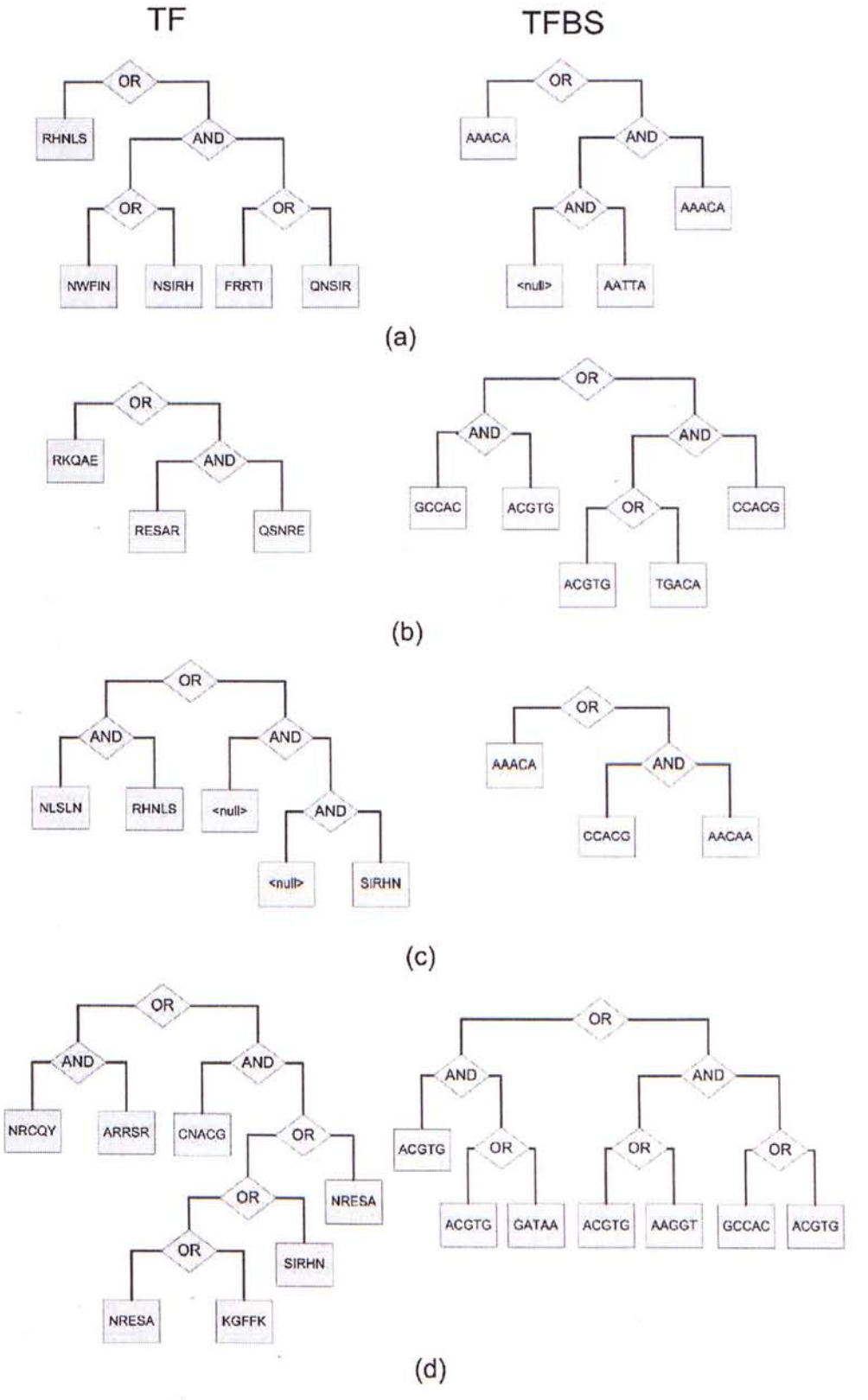


FIGURE A.1: Some examples in the generalized result

TABLE A.1: Number Of Pairs Discovered. Each integer denotes the number of pairs for each setting. The first row denotes the different settings for the value of k, whereas the first column denotes the minimal confidence (confidence level).

Confidence	TFBS4mer-TF4mer	TFBS4mer-TF5mer	TFBS4mer-TF6mer	TFBS5mer-TF4mer	TFBS5mer-TF5mer	TFBS5mer-TF6mer	TFBS6mer-TF4mer	TFBS6mer-TF5mer	TFBS6mer-TF6mer
0	4229	1334	871	420	262	200	100	74	59
0.1	3407	1155	761	414	262	200	100	74	59
0.2	710	435	294	354	240	184	95	74	59
0.3	260	163	115	264	180	134	82	72	59
0.4	28	15	4	170	126	92	52	44	36
0.5	2	0	0	94	70	48	22	20	16
0.6	0	0	0	34	24	10	0	12	14
0.7	0	0	0	6	6	2	0	0	0
0.8	0	0	0	0	0	0	0	0	0
0.9	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0

TABLE A.2: Number Of Pairs (duplicated pairs removed). Each integer denotes the number of pairs for each setting. The first row denotes the different settings for the value of k, whereas the first column denotes the minimal confidence (confidence level). Since some DNA k-mers are the reverse complements of themselves, the number of pairs are not necessary to be half of the corresponding number in Table A.1

Confidence	TFBS4mer-TF4mer	TFBS4mer-TF5mer	TFBS4mer-TF6mer	TFBS5mer-TF4mer	TFBS5mer-TF5mer	TFBS5mer-TF6mer	TFBS6mer-TF4mer	TFBS6mer-TF5mer	TFBS6mer-TF6mer
0.0	2153	689	452	210	131	100	56	42	34
0.1	1739	599	397	207	131	100	56	42	34
0.2	374	232	159	177	120	92	53	42	34
0.3	141	91	65	132	90	67	46	41	34
0.4	15	8	2	85	63	46	26	22	18
0.5	1	0	0	47	35	24	11	10	8
0.6	0	0	0	17	12	5	0	6	7
0.7	0	0	0	3	3	1	0	0	0
0.8	0	0	0	0	0	0	0	0	0
0.9	0	0	0	0	0	0	0	0	0
1.0	0	0	0	0	0	0	0	0	0

TABLE A.3: Number Of Merged Pairs. Each integer denotes the number of merged pairs for each setting. The first row denotes the different settings for the value of k, whereas the first column denotes the minimal confidence (confidence level). The number is significantly reduced as compared to that in Table A.2.

Confidence	TFBS4mer-TF4mer	TFBS4mer-TF5mer	TFBS4mer-TF6mer	TFBS5mer-TF4mer	TFBS5mer-TF5mer	TFBS5mer-TF6mer	TFBS6mer-TF4mer	TFBS6mer-TF5mer	TFBS6mer-TF6mer
0	1002	192	110	69	29	21	14	8	4
0.1	803	171	96	67	29	21	14	8	4
0.2	137	69	44	50	24	17	14	8	4
0.3	51	23	19	40	23	14	14	8	4
0.4	6	5	2	33	21	12	7	4	2
0.5	1	0	0	18	11	7	3	3	2
0.6	0	0	0	8	8	3	0	2	2
0.7	0	0	0	2	2	1	0	0	0
0.8	0	0	0	0	0	0	0	0	0
0.9	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0

TABLE A.4: Mean and SD of Support. Each block denotes the mean and SD of support of the pairs under different settings. The first row denotes the different settings for the value of k, whereas the first column denotes the minimal confidence (confidence level).

confidence	TFBS4mer-TF4mer	TFBS4mer-TF5mer	TFBS4mer-TF6mer	TFBS5mer-TF4mer	TFBS5mer-TF5mer	TFBS5mer-TF6mer	TFBS6mer-TF4mer	TFBS6mer-TF5mer	TFBS6mer-TF6mer
0.0	8.53±2.65	8.97±3.20	8.85±2.98	9.75±3.67	9.88±3.68	9.62±3.35	9.80±2.79	9.64±2.81	9.39±2.55
0.1	8.75±2.87	9.23±3.36	9.08±3.11	9.79±3.68	9.88±3.68	9.62±3.35	9.80±2.79	9.64±2.81	9.39±2.55
0.2	10.71±4.80	10.63±4.65	10.25±4.23	10.21±3.81	10.14±3.73	9.85±3.40	9.95±2.78	9.64±2.81	9.39±2.55
0.3	11.80±6.41	11.71±6.26	11.15±5.62	10.82±4.16	10.61±4.11	10.25±3.80	10.28±2.82	9.71±2.81	9.39±2.55
0.4	22.96±8.57	23.07±7.76	24.50±10.50	11.47±4.81	11.40±4.59	11.09±4.24	11.38±2.77	11.18±2.66	10.78±2.37
0.5	40.00±0.00	N/A	N/A	13.55±5.39	13.63±5.05	13.63±4.48	13.64±2.23	13.50±1.69	13.38±0.48
0.6	N/A	N/A	N/A	16.00±5.62	15.08±5.28	14.40±3.56	N/A	13.50±0.76	13.29±0.45
0.7	N/A	N/A	N/A	8.67±2.36	10.34±2.36	12.00±0.00	N/A	N/A	N/A
0.8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE A.5: Mean and SD of ϕ-coefficient. Each block denotes the mean and SD of ϕ-coefficient of the pairs under different settings. The first row denotes the different settings for the value of k, whereas the first column denotes the minimal confidence (confidence level).

Confidence	TFBS4mer-TF4mer	TFBS4mer-TF5mer	TFBS4mer-TF6mer	TFBS5mer-TF4mer	TFBS5mer-TF5mer	TFBS5mer-TF6mer	TFBS6mer-TF4mer	TFBS6mer-TF5mer	TFBS6mer-TF6mer
0.0	0.18±0.14	0.29±0.14	0.32±0.13	0.44±0.14	0.49±0.11	0.50±0.09	0.46±0.11	0.50±0.10	0.51±0.09
0.1	0.21±0.14	0.31±0.14	0.33±0.13	0.45±0.13	0.49±0.11	0.50±0.09	0.46±0.11	0.50±0.10	0.51±0.09
0.2	0.38±0.12	0.41±0.12	0.43±0.11	0.48±0.11	0.51±0.11	0.50±0.09	0.47±0.10	0.50±0.10	0.51±0.09
0.3	0.48±0.09	0.51±0.08	0.52±0.07	0.52±0.10	0.54±0.10	0.53±0.09	0.49±0.10	0.51±0.10	0.51±0.09
0.4	0.57±0.03	0.58±0.04	0.58±0.04	0.57±0.09	0.58±0.09	0.57±0.08	0.53±0.08	0.56±0.08	0.55±0.07
0.5	0.57±0.00	N/A	N/A	0.62±0.08	0.64±0.07	0.63±0.05	0.60±0.04	0.63±0.03	0.62±0.02
0.6	N/A	N/A	N/A	0.70±0.06	0.71±0.06	0.69±0.04	N/A	0.62±0.02	0.62±0.02
0.7	N/A	N/A	N/A	0.80±0.04	0.79±0.03	0.77±0.00	N/A	N/A	N/A
0.8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE A.6: Mean and SD of Lift. Each block denotes the mean and SD of lift of the pairs under different settings. The first row denotes the different settings for the value of k, whereas the first column denotes the minimal confidence (confidence level).

Confidence	TFBS4mer-TF4mer	TFBS4mer-TF5mer	TFBS4mer-TF6mer	TFBS5mer-TF4mer	TFBS5mer-TF5mer	TFBS5mer-TF6mer	TFBS6mer-TF4mer	TFBS6mer-TF5mer	TFBS6mer-TF6mer
0.0	5.09±4.80	8.67±6.13	9.56±6.16	15.29±8.23	17.92±7.34	18.14±5.80	14.98±4.45	17.96±3.60	18.54±2.96
0.1	5.71±5.09	9.28±6.31	10.13±6.37	15.50±8.11	17.92±7.34	18.14±5.80	14.98±4.45	17.96±3.60	18.54±2.96
0.2	11.71±6.96	13.49±7.64	14.37±7.66	16.60±8.11	18.32±7.46	18.43±5.87	15.42±4.07	17.96±3.60	18.54±2.96
0.3	16.25±7.55	18.67±8.12	19.40±7.92	18.33±8.53	19.81±7.79	19.72±6.04	15.89±3.90	18.00±3.65	18.54±2.96
0.4	11.56±5.37	10.96±4.22	11.97±5.84	20.97±9.30	21.41±8.53	20.87±6.67	16.67±3.31	18.75±3.28	19.12±2.39
0.5	5.94±0.00	N/A	N/A	21.97±11.37	22.57±10.46	21.41±8.05	18.15±3.50	19.37±3.23	18.95±0.66
0.6	N/A	N/A	N/A	24.62±15.14	25.80±13.76	25.49±8.90	N/A	18.75±0.63	19.02±0.67
0.7	N/A	N/A	N/A	51.93±15.80	42.07±14.87	31.53±0.00	N/A	N/A	N/A
0.8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE A.7: Mean and SD of Forward Conviction. Each block denotes the mean and SD of forward conviction of the pairs under different settings. The first row denotes the different settings for the value of k, whereas the first column denotes the minimal confidence (confidence level).

Confidence	TFBS4mer-TF4mer	TFBS4mer-TF5mer	TFBS4mer-TF6mer	TFBS5mer-TF4mer	TFBS5mer-TF5mer	TFBS5mer-TF6mer	TFBS6mer-TF4mer	TFBS6mer-TF5mer	TFBS6mer-TF6mer
0.0	1.15±0.14	1.21±0.16	1.21±0.15	1.89±0.71	1.89±0.67	1.77±0.53	2.39±1.15	2.33±1.02	2.12±0.58
0.1	1.17±0.14	1.23±0.16	1.23±0.15	1.90±0.71	1.89±0.67	1.77±0.53	2.39±1.15	2.33±1.02	2.12±0.58
0.2	1.38±0.16	1.38±0.16	1.38±0.16	1.97±0.70	1.94±0.68	1.81±0.53	2.42±1.17	2.33±1.02	2.12±0.58
0.3	1.52±0.11	1.53±0.09	1.52±0.09	2.08±0.68	2.02±0.64	1.90±0.47	2.44±1.25	2.32±1.03	2.12±0.58
0.4	1.76±0.10	1.75±0.06	1.78±0.01	2.33±0.72	2.21±0.66	2.06±0.47	2.53±1.25	2.40±1.05	2.18±0.59
0.5	2.04±0.00	N/A	N/A	2.58±0.73	2.49±0.70	2.26±0.51	3.33±1.49	3.08±1.16	2.79±0.33
0.6	N/A	N/A	N/A	3.34±0.62	3.33±0.57	3.17±0.40	N/A	2.79±0.36	2.73±0.31
0.7	N/A	N/A	N/A	3.50±0.28	3.70±0.29	3.90±0.00	N/A	N/A	N/A
0.8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE A.8: Mean and SD of Backward Conviction. Each block denotes the mean and SD of backward conviction of the pairs under different settings. The first row denotes the different settings for the value of k, whereas the first column denotes the minimal confidence (confidence level).

Confidence	TFBS4mer-TF4mer	TFBS4mer-TF5mer	TFBS4mer-TF6mer	TFBS5mer-TF4mer	TFBS5mer-TF5mer	TFBS5mer-TF6mer	TFBS6mer-TF4mer	TFBS6mer-TF5mer	TFBS6mer-TF6mer
0.0	2.04±1.73	3.44±2.62	4.01±2.70	2.61±1.61	3.50±2.29	3.71±2.15	1.96±0.86	2.46±1.55	2.45±0.83
0.1	2.14±1.83	3.52±2.69	4.07±2.79	2.64±1.61	3.50±2.29	3.71±2.15	1.96±0.86	2.46±1.55	2.45±0.83
0.2	3.17±2.33	4.02±3.05	4.30±3.05	2.72±1.64	3.51±2.30	3.70±2.14	2.00±0.86	2.46±1.55	2.45±0.83
0.3	4.28±2.55	5.99±3.15	6.44±3.14	2.80±1.58	3.46±2.31	3.44±2.02	2.10±0.88	2.49±1.57	2.45±0.83
0.4	4.90±1.99	5.87±3.17	5.28±2.33	2.87±1.30	3.61±2.40	3.68±2.35	2.29±0.89	2.90±1.79	2.74±0.64
0.5	2.75±0.00	N/A	N/A	3.13±1.25	4.35±2.65	4.85±2.72	2.64±1.17	3.25±2.47	2.54±0.10
0.6	N/A	N/A	N/A	3.71±1.63	4.21±2.55	3.50±0.76	N/A	2.54±0.07	2.57±0.06
0.7	N/A	N/A	N/A	3.90±0.00	4.87±0.00	4.87±0.00	N/A	N/A	N/A
0.8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE A.9: Mean and SD of TFBS Prediction Score. Each block denotes the mean and SD of TFBS Prediction Score of the pairs under different settings. The first row denotes the different settings for the value of k, whereas the first column denotes the minimal confidence (confidence level).

Confidence	TFBS4mer-TF4mer	TFBS4mer-TF5mer	TFBS4mer-TF6mer	TFBS5mer-TF4mer	TFBS5mer-TF5mer	TFBS5mer-TF6mer	TFBS6mer-TF4mer	TFBS6mer-TF5mer	TFBS6mer-TF6mer
0.0	0.29±0.35	0.49±0.40	0.52±0.42	0.36±0.36	0.41±0.41	0.40±0.43	0.21±0.33	0.18±0.33	0.13±0.30
0.1	0.31±0.36	0.47±0.41	0.50±0.42	0.37±0.36	0.41±0.41	0.40±0.43	0.21±0.33	0.18±0.33	0.13±0.30
0.2	0.40±0.36	0.46±0.41	0.44±0.43	0.39±0.36	0.40±0.41	0.38±0.42	0.22±0.34	0.18±0.33	0.13±0.30
0.3	0.45±0.39	0.49±0.43	0.50±0.44	0.41±0.35	0.38±0.39	0.36±0.40	0.24±0.35	0.18±0.34	0.13±0.30
0.4	0.56±0.37	0.63±0.38	0.68±0.13	0.45±0.35	0.38±0.40	0.33±0.40	0.38±0.39	0.33±0.40	0.25±0.37
0.5	0.73±0.00	N/A	N/A	0.39±0.37	0.42±0.40	0.42±0.42	0.34±0.40	0.34±0.42	0.31±0.41
0.6	N/A	N/A	N/A	0.53±0.38	0.62±0.37	0.83±0.09	N/A	0.14±0.31	0.24±0.38
0.7	N/A	N/A	N/A	0.30±0.43	0.63±0.45	0.95±0.00	N/A	N/A	N/A
0.8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE A.10: Mean and SD of TFBS Binding Prediction Score. Each block denotes the mean and SD of TFBS Binding Prediction Score of the pairs under different settings. The first row denotes the different settings for the value of k, whereas the first column denotes the minimal confidence (confidence level).

Confidence	TFBS4mer-TF4mer	TFBS4mer-TF5mer	TFBS4mer-TF6mer	TFBS5mer-TF4mer	TFBS5mer-TF5mer	TFBS5mer-TF6mer	TFBS6mer-TF4mer	TFBS6mer-TF5mer	TFBS6mer-TF6mer
0.0	0.06±0.16	0.20±0.29	0.26±0.32	0.15±0.23	0.24±0.30	0.25±0.33	0.11±0.20	0.12±0.24	0.12±0.26
0.1	0.07±0.16	0.20±0.28	0.24±0.31	0.15±0.23	0.24±0.30	0.25±0.33	0.11±0.20	0.12±0.24	0.12±0.26
0.2	0.14±0.23	0.22±0.29	0.23±0.32	0.15±0.21	0.23±0.29	0.24±0.32	0.12±0.21	0.12±0.24	0.12±0.26
0.3	0.16±0.25	0.22±0.31	0.25±0.34	0.18±0.23	0.25±0.29	0.27±0.34	0.13±0.22	0.12±0.24	0.12±0.26
0.4	0.38±0.27	0.47±0.34	0.75±0.25	0.21±0.25	0.25±0.31	0.24±0.33	0.22±0.25	0.23±0.29	0.23±0.32
0.5	0.55±0.00	N/A	N/A	0.23±0.27	0.30±0.33	0.33±0.37	0.23±0.28	0.27±0.33	0.27±0.35
0.6	N/A	N/A	N/A	0.39±0.29	0.50±0.31	0.68±0.11	N/A	0.13±0.28	0.22±0.35
0.7	N/A	N/A	N/A	0.24±0.34	0.50±0.35	0.75±0.00	N/A	N/A	N/A
0.8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE A.11: Mean and SD of Binding Prediction Score. Each block denotes the mean and SD of Binding Prediction Score of the pairs under different settings. The first row denotes the different settings for the value of k, whereas the first column denotes the minimal confidence (confidence level).

Confidence	TFBS4mer-TF4mer	TFBS4mer-TF5mer	TFBS4mer-TF6mer	TFBS5mer-TF4mer	TFBS5mer-TF5mer	TFBS5mer-TF6mer	TFBS6mer-TF4mer	TFBS6mer-TF5mer	TFBS6mer-TF6mer
0.0	0.13±0.28	0.30±0.37	0.35±0.39	0.28±0.36	0.36±0.39	0.36±0.41	0.16±0.27	0.15±0.29	0.16±0.34
0.1	0.14±0.29	0.31±0.37	0.34±0.39	0.29±0.36	0.36±0.39	0.36±0.41	0.16±0.27	0.15±0.29	0.16±0.34
0.2	0.26±0.34	0.31±0.37	0.32±0.38	0.30±0.36	0.35±0.39	0.35±0.41	0.16±0.28	0.15±0.29	0.16±0.34
0.3	0.26±0.35	0.29±0.37	0.33±0.39	0.31±0.35	0.35±0.39	0.37±0.42	0.17±0.28	0.15±0.30	0.16±0.34
0.4	0.50±0.33	0.56±0.36	0.83±0.17	0.34±0.36	0.34±0.40	0.31±0.40	0.30±0.32	0.28±0.36	0.30±0.42
0.5	0.75±0.00	N/A	N/A	0.33±0.38	0.39±0.42	0.39±0.43	0.30±0.34	0.31±0.40	0.32±0.42
0.6	N/A	N/A	N/A	0.51±0.36	0.61±0.37	0.82±0.08	N/A	0.15±0.34	0.26±0.42
0.7	N/A	N/A	N/A	0.26±0.37	0.53±0.37	0.79±0.00	N/A	N/A	N/A
0.8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE A.12: Mean and SD of TFBS Prediction Score (Merged Pairs). Each block denotes the mean and SD of TFBS Prediction Score of the merged pairs under different settings. The first row denotes the different settings for the value of k, whereas the first column denotes the minimal confidence (confidence level).

Confidence	TFBS4mer-TF4mer	TFBS4mer-TF5mer	TFBS4mer-TF6mer	TFBS5mer-TF4mer	TFBS5mer-TF5mer	TFBS5mer-TF6mer	TFBS6mer-TF4mer	TFBS6mer-TF5mer	TFBS6mer-TF6mer
0.0	0.20±0.34	0.38±0.40	0.44±0.42	0.34±0.38	0.47±0.43	0.50±0.45	0.29±0.35	0.34±0.40	0.14±0.24
0.1	0.22±0.35	0.40±0.40	0.41±0.42	0.35±0.38	0.47±0.43	0.50±0.45	0.29±0.35	0.34±0.40	0.14±0.24
0.2	0.38±0.38	0.47±0.40	0.46±0.42	0.35±0.35	0.39±0.40	0.43±0.43	0.29±0.35	0.34±0.40	0.14±0.24
0.3	0.39±0.39	0.40±0.41	0.39±0.42	0.41±0.35	0.43±0.40	0.38±0.41	0.27±0.36	0.34±0.40	0.14±0.24
0.4	0.72±0.33	0.67±0.35	0.88±0.13	0.42±0.34	0.36±0.37	0.33±0.38	0.44±0.37	0.61±0.39	0.27±0.27
0.5	0.73±0.00	N/A	N/A	0.36±0.30	0.40±0.34	0.44±0.39	0.60±0.43	0.60±0.43	0.42±0.42
0.6	N/A	N/A	N/A	0.45±0.40	0.51±0.40	0.82±0.12	N/A	0.42±0.42	0.42±0.42
0.7	N/A	N/A	N/A	0.45±0.45	0.48±0.48	0.95±0.00	N/A	N/A	N/A
0.8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE A.13: Mean and SD of TFBS Binding Prediction Score (Merged Pairs). Each block denotes the mean and SD of TFBS Binding Prediction Score of the merged pairs under different settings. The first row denotes the different settings for the value of k, whereas the first column denotes the minimal confidence (confidence level).

Confidence	TFBS4mer-TF4mer	TFBS4mer-TF5mer	TFBS4mer-TF6mer	TFBS5mer-TF4mer	TFBS5mer-TF5mer	TFBS5mer-TF6mer	TFBS6mer-TF4mer	TFBS6mer-TF5mer	TFBS6mer-TF6mer
0.0	0.04±0.15	0.17±0.29	0.24±0.34	0.17±0.30	0.31±0.37	0.37±0.39	0.14±0.20	0.18±0.24	0.14±0.24
0.1	0.05±0.16	0.19±0.30	0.24±0.34	0.18±0.30	0.31±0.37	0.37±0.39	0.14±0.20	0.18±0.24	0.14±0.24
0.2	0.16±0.26	0.24±0.31	0.24±0.33	0.17±0.26	0.25±0.32	0.29±0.35	0.14±0.20	0.18±0.24	0.14±0.24
0.3	0.19±0.30	0.24±0.35	0.26±0.38	0.22±0.27	0.31±0.34	0.34±0.39	0.13±0.20	0.18±0.24	0.14±0.24
0.4	0.54±0.32	0.52±0.35	0.75±0.25	0.23±0.26	0.26±0.28	0.26±0.32	0.25±0.23	0.37±0.22	0.27±0.27
0.5	0.55±0.00	N/A	N/A	0.21±0.28	0.30±0.31	0.30±0.36	0.44±0.33	0.44±0.33	0.39±0.39
0.6	N/A	N/A	N/A	0.40±0.36	0.44±0.35	0.68±0.13	N/A	0.38±0.38	0.39±0.39
0.7	N/A	N/A	N/A	0.36±0.36	0.38±0.38	0.75±0.00	N/A	N/A	N/A
0.8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE A.14: Mean and SD of Binding Prediction Score (Merged Pairs). Each block denotes the mean and SD of Binding Prediction Score of the merged pairs under different settings. The first row denotes the different settings for the value of k, whereas the first column denotes the minimal confidence (confidence level).

Confidence	TFBS4mer-TF4mer	TFBS4mer-TF5mer	TFBS4mer-TF6mer	TFBS5mer-TF4mer	TFBS5mer-TF5mer	TFBS5mer-TF6mer	TFBS6mer-TF4mer	TFBS6mer-TF5mer	TFBS6mer-TF6mer
0.0	0.07±0.22	0.25±0.38	0.33±0.41	0.26±0.38	0.44±0.44	0.46±0.43	0.25±0.34	0.25±0.35	0.25±0.43
0.1	0.08±0.24	0.27±0.38	0.34±0.41	0.27±0.39	0.44±0.44	0.46±0.43	0.25±0.34	0.25±0.35	0.25±0.43
0.2	0.25±0.37	0.33±0.39	0.32±0.40	0.32±0.39	0.40±0.42	0.42±0.43	0.25±0.34	0.25±0.35	0.25±0.43
0.3	0.30±0.40	0.35±0.42	0.40±0.44	0.42±0.41	0.47±0.45	0.49±0.45	0.19±0.31	0.25±0.35	0.25±0.43
0.4	0.61±0.34	0.61±0.37	0.83±0.17	0.39±0.39	0.41±0.43	0.39±0.42	0.37±0.36	0.50±0.36	0.50±0.50
0.5	0.75±0.00	N/A	N/A	0.35±0.41	0.46±0.44	0.37±0.43	0.49±0.38	0.49±0.38	0.47±0.47
0.6	N/A	N/A	N/A	0.54±0.42	0.54±0.42	0.82±0.08	N/A	0.45±0.45	0.47±0.47
0.7	N/A	N/A	N/A	0.39±0.39	0.39±0.39	0.79±0.00	N/A	N/A	N/A
0.8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE A.15: Number of Pairs with at least one related PDB chain. Each integer denotes the number of pairs with at least one related PDB chain for each setting. The first row denotes the different settings for the value of k, whereas the first column denotes the minimal confidence (confidence level)

Confidence	TFBS4mer-TF4mer	TFBS4mer-TF5mer	TFBS4mer-TF6mer	TFBS5mer-TF4mer	TFBS5mer-TF5mer	TFBS5mer-TF6mer	TFBS6mer-TF4mer	TFBS6mer-TF5mer	TFBS6mer-TF6mer
0.0	1347	500	323	151	80	55	19	10	6
0.1	1140	427	274	150	80	55	19	10	6
0.2	283	162	101	131	71	48	18	10	6
0.3	102	62	44	95	50	31	16	10	6
0.4	11	6	2	62	32	20	14	9	6
0.5	1	N/A	N/A	29	19	12	5	4	3
0.6	N/A	N/A	N/A	12	9	5	N/A	1	2
0.7	N/A	N/A	N/A	1	2	1	N/A	N/A	N/A
0.8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE A.16: Number of Pairs with at least one related PDB chain (Merged Pairs). Each integer denotes the number of merged pairs with at least one related PDB chain for each setting. The first row denotes the different settings for the value of k, whereas the first column denotes the minimal confidence (confidence level)

Confidence	TFBS4mer-TF4mer	TFBS4mer-TF5mer	TFBS4mer-TF6mer	TFBS5mer-TF4mer	TFBS5mer-TF5mer	TFBS5mer-TF6mer	TFBS6mer-TF4mer	TFBS6mer-TF5mer	TFBS6mer-TF6mer
0.0	379	112	69	46	19	14	7	4	1
0.1	337	106	58	46	19	14	7	4	1
0.2	90	49	30	35	15	10	7	4	1
0.3	34	15	13	30	15	8	6	4	1
0.4	5	4	2	24	12	6	5	3	1
0.5	1	N/A	N/A	14	7	4	2	2	1
0.6	N/A	N/A	N/A	5	5	3	N/A	1	1
0.7	N/A	N/A	N/A	1	1	1	N/A	N/A	N/A
0.8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE A.17: Fraction of Verified Pairs. Each fraction denotes the fraction of verified pairs for each setting. The first row denotes the different settings for the value of k, whereas the first column denotes the minimal confidence (confidence level)

Confidence	TFBS4mer-TF4mer	TFBS4mer-TF5mer	TFBS4mer-TF6mer	TFBS5mer-TF4mer	TFBS5mer-TF5mer	TFBS5mer-TF6mer	TFBS6mer-TF4mer	TFBS6mer-TF5mer	TFBS6mer-TF6mer
0.0	0.3942094	0.684	0.7585139	0.6225166	0.8125	0.8363636	0.7894737	0.9	1
0.1	0.4184211	0.7189696	0.7810219	0.6266667	0.8125	0.8363636	0.7894737	0.9	1
0.2	0.6466431	0.7530864	0.7920792	0.6641221	0.8309859	0.8333333	0.8333333	0.9	1
0.3	0.6568627	0.7258065	0.75	0.7263158	0.88	0.9393939	0.875	0.9	1
0.4	1	1	1	0.7419355	0.875	0.9	0.9285714	1	1
0.5	1	N/A	N/A	0.7586207	0.8947368	0.9166667	1	1	1
0.6	N/A	N/A	N/A	1	1	1	N/A	1	1
0.7	N/A	N/A	N/A	1	1	1	N/A	N/A	N/A
0.8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE A.18: Fraction of Verified Pairs (Merged Pairs). Each fraction denotes the fraction of merged pairs verified for each setting. The first row denotes the different settings for the value of k, whereas the first column denotes the minimal confidence (confidence level)

Confidence	TFBS4mer-TF4mer	TFBS4mer-TF5mer	TFBS4mer-TF6mer	TFBS5mer-TF4mer	TFBS5mer-TF5mer	TFBS5mer-TF6mer	TFBS6mer-TF4mer	TFBS6mer-TF5mer	TFBS6mer-TF6mer
0.0	0.3271768	0.6696429	0.7391304	0.5652174	0.8421053	0.8571429	0.8571429	0.75	1
0.1	0.3560831	0.6981132	0.7931034	0.5652174	0.8421053	0.8571429	0.8571429	0.75	1
0.2	0.6444444	0.7755102	0.7333333	0.6857143	0.8666667	0.9	0.8571429	0.75	1
0.3	0.6470588	0.8	0.7692308	0.8	0.8666667	1	0.8333333	0.75	1
0.4	1	1	1	0.7916667	0.9166667	1	0.8	1	1
0.5	1	N/A	N/A	0.5714286	0.8571429	0.75	1	1	1
0.6	N/A	N/A	N/A	1	1	1	N/A	1	1
0.7	N/A	N/A	N/A	1	1	1	N/A	N/A	N/A
0.8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE A.19: Maximal TFBS Prediction Score over 50 runs. Each block denotes the maximal performance of TFBS Prediction Score of the pairs over 50 random runs under different settings. The first row denotes the different settings for the value of k, whereas the first column denotes the minimal confidence (confidence level).

Confidence	TFBS4mer-TF4mer	TFBS4mer-TF5mer	TFzBS4mer-TF6mer	TFBS5mer-TF4mer	TFBS5mer-TF5mer	TFBS5mer-TF6mer	TFBS6mer-TF4mer	TFBS6mer-TF5mer	TFBS6mer-TF6mer
0.0	0.14469±0.3095	0.09185±0.25819	0.04296±0.17842	0.05925±0.2266	0.03767±0.18401	0.02667±0.15762	0.04292±0.19751	0.04678±0.19528	0.01695±0.12908
0.1	0.15028±0.3172	0.09067±0.25405	0.04943±0.19171	0.0558±0.22102	0.04018±0.19693	0.02375±0.14916	0.05356±0.20521	0.02815±0.16226	0.02018±0.13098
0.2	0.16204±0.32886	0.10662±0.27429	0.06548±0.22314	0.06866±0.23892	0.03588±0.1824	0.02222±0.14515	0.03775±0.18377	0.02703±0.16216	0.01695±0.12908
0.3	0.17702±0.34032	0.10803±0.28737	0.07302±0.23572	0.08541±0.26954	0.04931±0.2151	0.02456±0.14834	0.05358±0.20581	0.02381±0.14286	0.02119±0.13251
0.4	0.28267±0.38034	0.25946±0.37558	0.25±0.43301	0.06072±0.22946	0.06365±0.23856	0.02998±0.16459	0.05769±0.23316	0.03409±0.16507	0.02778±0.16434
0.5	0.87037±0.12963	N/A	N/A	0.078±0.25463	0.05713±0.20329	0.04167±0.19983	0.06818±0.22841	0.05±0.21794	0.0625±0.24206
0.6	N/A	N/A	N/A	0.14163±0.30753	0.06483±0.22311	0.1±0.3	N/A	0.08333±0.27639	0.07143±0.25754
0.7	N/A	N/A	N/A	0.23333±0.37268	0.16667±0.37268	0±0	N/A	N/A	N/A
0.8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE A.20: Maximal TFBS Binding Prediction Score over 50 runs. Each block denotes the maximal performance of TFBS Binding Prediction Score of the pairs over 50 random runs under different settings. The first row denotes the different settings for the value of k, whereas the first column denotes the minimal confidence (confidence level).

Confidence	TFBS4mer-TF4mer	TFBS4mer-TF5mer	TFBS4mer-TF6mer	TFBS5mer-TF4mer	TFBS5mer-TF5mer	TFBS5mer-TF6mer	TFBS6mer-TF4mer	TFBS6mer-TF5mer	TFBS6mer-TF6mer
0.0	0.00332±0.04768	0.00222±0.03928	0.00115±0.03386	0.00516±0.06929	0.00382±0.06166	0.005±0.07053	0.01549±0.11083	0.01351±0.11546	0.01695±0.12908
0.1	0.00321±0.04947	0.00247±0.04357	0.00131±0.03623	0.00631±0.07446	0.00763±0.08704	0±0	0.01634±0.11728	0.0045±0.03849	0±0
0.2	0.00496±0.05402	0.00374±0.04772	0.00612±0.07444	0.00857±0.09168	0.00729±0.08036	0.00514±0.07352	0.02456±0.14703	0.01351±0.11546	0.01695±0.12908
0.3	0.00751±0.07192	0.00241±0.02201	0.005±0.04689	0.00821±0.08726	0.00556±0.07433	0±0	0.01829±0.1221	0.01389±0.11703	0.00254±0.01936
0.4	0.03571±0.18558	0.00667±0.02494	0±0	0.00588±0.07647	0.00794±0.08873	0.00036±0.00342	0.02404±0.14091	0.01136±0.07452	0.00185±0.01096
0.5	0.5±0.5	N/A	N/A	0.01064±0.10259	0.01429±0.11867	0.0026±0.01785	0.02273±0.10415	0.03333±0.1453	0±0
0.6	N/A	N/A	N/A	0.02941±0.16896	0.02083±0.09991	0±0	N/A	0±0	0±0
0.7	N/A	N/A	N/A	0.11111±0.24845	0±0	0±0	N/A	N/A	N/A
0.8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE A.21: Maximal Binding Prediction Score over 50 runs. Each block denotes the maximal performance of Binding Prediction Score of the pairs over 50 random runs under different settings. The first row denotes the different settings for the value of k, whereas the first column denotes the minimal confidence (confidence level).

Confidence	TFBS4mer-TF4mer	TFBS4mer-TF5mer	TFBS4mer-TF6mer	TFBS5mer-TF4mer	TFBS5mer-TF5mer	TFBS5mer-TF6mer	TFBS6mer-TF4mer	TFBS6mer-TF5mer	TFBS6mer-TF6mer
0.0	0.0073±0.0802	0.00354±0.05666	0.00237±0.04362	0.00774±0.07773	0.00509±0.06492	0.0035±0.07053	0.025±0.1479	0.02703±0.16216	0.01695±0.12908
0.1	0.00741±0.07814	0.00394±0.06054	0.00197±0.04048	0.00966±0.09144	0.00763±0.08704	0±0	0.01812±0.12757	0.01351±0.11546	0±0
0.2	0.01156±0.09813	0.00736±0.07301	0.0068±0.0822	0.0096±0.09399	0.0099±0.09392	0.00543±0.07352	0.01158±0.17488	0.01351±0.11546	0.01695±0.12908
0.3	0.01517±0.11226	0.01361±0.10318	0.01304±0.10344	0.00947±0.09183	0.00926±0.0891	0±0	0.02439±0.15426	0.02083±0.1301	0.00847±0.06454
0.4	0.07143±0.25754	0.06667±0.24944	0±0	0.01471±0.1141	0.00791±0.08873	0.00047±0.00451	0.03846±0.19231	0.02273±0.14903	0.02778±0.16434
0.5	0.5±0.5	N/A	N/A	0.01463±0.10918	0.01429±0.11867	0.01042±0.07141	0.04545±0.2083	0.05±0.21794	0±0
0.6	N/A	N/A	N/A	0.03088±0.14039	0.04167±0.19983	0±0	N/A	0±0	0±0
0.7	N/A	N/A	N/A	0.11111±0.24845	0±0	0±0	N/A	N/A	N/A
0.8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE A.22: Maximal TFBS Prediction Score over 50 runs (Merged Pairs). Each block denotes the maximal performance of TFBS Prediction Score of the merged pairs over 50 random runs under different settings. The first row denotes the different settings for the value of k, whereas the first column denotes the minimal confidence (confidence level).

Confidence	TFBS4mer-TF4mer	TFBS4mer-TF5mer	TFBS4mer-TF6mer	TFBS5mer-TF4mer	TFBS5mer-TF5mer	TFBS5mer-TF6mer	TFBS6mer-TF4mer	TFBS6mer-TF5mer	TFBS6mer-TF6mer
0.0	0.14387±0.3094	0.09185±0.25819	0.04301±0.17852	0.05925±0.2266	0.03767±0.18401	0.02667±0.15762	0.04292±0.19751	0.04678±0.19528	0.01695±0.12908
0.1	0.15047±0.31761	0.09067±0.25405	0.04943±0.19171	0.0558±0.22102	0.04018±0.19093	0.02375±0.14916	0.05456±0.20521	0.02815±0.16226	0.02018±0.13098
0.2	0.16227±0.32903	0.10662±0.27429	0.06548±0.22314	0.06866±0.23892	0.03588±0.1824	0.02222±0.14515	0.03775±0.18377	0.02703±0.16216	0.01695±0.12908
0.3	0.17702±0.34032	0.10803±0.28737	0.07302±0.23572	0.08541±0.26954	0.04931±0.2151	0.02456±0.14834	0.05358±0.20581	0.02381±0.14286	0.02119±0.13251
0.4	0.28267±0.38034	0.25946±0.37558	0.25±0.43301	0.06072±0.22946	0.06365±0.23856	0.02998±0.16459	0.05769±0.23316	0.03409±0.16507	0.02778±0.16434
0.5	0.87037±0.12963	N/A	N/A	0.078±0.25463	0.05713±0.20329	0.04167±0.19983	0.06818±0.22841	0.05±0.21794	0.0625±0.24206
0.6	N/A	N/A	N/A	0.14163±0.30753	0.06483±0.22311	0.1±0.3	N/A	0.08333±0.27639	0.07143±0.25754
0.7	N/A	N/A	N/A	0.23333±0.37268	0.16667±0.37268	0±0	N/A	N/A	N/A
0.8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE A.23: Maximal TFBS Binding Prediction Score over 50 runs (Merged Pairs). Each block denotes the maximal performance of TFBS Binding Prediction Score of the merged pairs over 50 random runs under different settings. The first row denotes the different settings for the value of k, whereas the first column denotes the minimal confidence (confidence level).

	TFBS4mer-TF4mer	TFBS4mer-TF5mer	TFBS4mer-TF6mer	TFBS5mer-TF4mer	TFBS5mer-TF5mer	TFBS5mer-TF6mer	TFBS6mer-TF4mer	TFBS6mer-TF5mer	TFBS6mer-TF6mer
0.0	0.00334±0.04777	0.00222±0.03928	0.00115±0.03386	0.00516±0.06929	0.00382±0.06166	0.0035±0.07053	0.01564±0.11738	0.01351±0.11546	0.01695±0.12908
0.1	0.00322±0.04956	0.00247±0.04357	0.00131±0.03623	0.00631±0.07446	0.00763±0.08704	0±0	0.01634±0.11728	0.0045±0.03849	0±0
0.2	0.00496±0.05402	0.00374±0.04772	0.00612±0.07444	0.00857±0.09168	0.00729±0.08036	0.00543±0.07352	0.02456±0.14703	0.01351±0.11546	0.01695±0.12908
0.3	0.00751±0.07192	0.00241±0.02201	0.0035±0.04689	0.00821±0.08726	0.00556±0.07433	0±0	0.01829±0.1221	0.01389±0.11703	0.00254±0.01936
0.4	0.03571±0.18558	0.00667±0.02494	0±0	0.00588±0.07647	0.00794±0.08873	0.00036±0.00342	0.02404±0.14091	0.01136±0.07452	0.00185±0.01096
0.5	0.5±0.5	N/A	N/A	0.01064±0.10259	0.01429±0.11867	0.0026±0.01785	0.02273±0.10415	0.03333±0.1453	0±0
0.6	N/A	N/A	N/A	0.02941±0.16896	0.02083±0.09991	0±0	N/A	0±0	0±0
0.7	N/A	N/A	N/A	0.11111±0.24845	0±0	0±0	N/A	N/A	N/A
0.8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE A.24: Maximal Binding Prediction Score over 50 runs (Merged Pairs). Each block denotes the maximal performance of Binding Prediction Score of the merged pairs over 50 random runs under different settings. The first row denotes the different settings for the value of k, whereas the first column denotes the minimal confidence (confidence level).

	TFBS4mer-TF4mer	TFBS4mer-TF5mer	TFBS4mer-TF6mer	TFBS5mer-TF4mer	TFBS5mer-TF5mer	TFBS5mer-TF6mer	TFBS6mer-TF4mer	TFBS6mer-TF5mer	TFBS6mer-TF6mer
0.0	0.00709±0.07892	0.00354±0.05666	0.00237±0.04365	0.00774±0.07773	0.00509±0.06492	0.0035±0.07053	0.02525±0.14863	0.02703±0.16216	0.01695±0.12908
0.1	0.00743±0.07824	0.00394±0.06054	0.00197±0.04048	0.00969±0.09155	0.00763±0.08704	0±0	0.01812±0.12757	0.01351±0.11546	0±0
0.2	0.01156±0.09813	0.00736±0.07301	0.0068±0.0822	0.0096±0.09399	0.0099±0.09392	0.00543±0.07352	0.01158±0.17488	0.01351±0.11546	0.01695±0.12908
0.3	0.01517±0.11226	0.01361±0.10318	0.01304±0.10344	0.00947±0.09183	0.00926±0.0891	0±0	0.02439±0.15426	0.02083±0.1301	0.00847±0.06454
0.4	0.07143±0.25754	0.06667±0.24944	0±0	0.01471±0.1141	0.00791±0.08873	0.00047±0.00451	0.03846±0.19231	0.02273±0.14903	0.02778±0.16434
0.5	0.5±0.5	N/A	N/A	0.01463±0.10918	0.01429±0.11867	0.01042±0.07141	0.04545±0.2083	0.05±0.21794	0±0
0.6	N/A	N/A	N/A	0.03088±0.14039	0.04167±0.19983	0±0	N/A	0±0	0±0
0.7	N/A	N/A	N/A	0.11111±0.24845	0±0	0±0	N/A	N/A	N/A
0.8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE A.25: 631 TRANSFAC 2008.3 IDs and Factor Names used

ID	Factor Name	ID	Factor Name	ID	Factor Name	ID	Factor Name	ID	Factor Name	ID	Factor Name
T00003	AS-C T3	T00842	Tra-1 (long form)	T01950	HNF-1alpha-B	T04378	Mad	T08676	STAT6	T09986	NF-AT4
T00008	Adf-1	T00843	Ttk 69K	T01951	HNF-1alpha-C	T04446	Nkx5-1	T08787	ARF1 isoform-1	T09990	CDP-isoform1
T00011	ADR1	T00851	T3R-beta1	T01973	REST-form2	T04539	RPN4	T08797	MCB1	T10028	SREBP-1c
T00019	AhR	T00863	Ubx	T01992	Abd-A	T04610	SXR	T08805	WRKY1	T10030	POU3F2
T00026	Antp	T00886	v-ErbA	T02003	Cdx-3	T04651	ER-beta	T08823	E2F	T10059	GCMa
T00028	YAP1	T00891	HNF-1beta-A	T02008	Ems	T04665	Xvent-1	T08853	myogenin	T10068	COUP-TF2
T00033	AP-2alpha	T00893	v-Jun	T02030	Sd	T04674	IRF-7A	T08858	REVERB-alpha	T10083	HNF-3alpha
T00063	Rcd	T00894	Vnw65	T02033	HsfA1	T04675	MRF-2-isoform1	T08863	S8	**T10144**	**Gfi1b**
T00077	CACCC-binding factor	T00895	v-Myb	T02039	HAC1	T04679	dri	T08868	CTCF	T10187	NF-E2 p45
T00079	Cad	T00899	WT1	T02050	Nkx6-2	T04728	CDC5L	T08878	Opaque-2	T10207	GATA-6
T00080	CBF1	T00910	YB-1	T02054	HOX11	**T04733**	**Alfin1**	T08972	EAR2	T10209	Nkx2-1
T00104	C/EBPalpha	T00915	YY1	T02063	KNOX3	T04734	Topors-isoform1	T08978	Dl-A	T10211	Evi-1
T00106	C/EBP	T00917	Zen-1	T02068	PU.1	T04783	mtTFA	T08985	Pit4	T10265	LRH-1
T00109	C/EBPdelta	T00918	Zeste	T02099	Zen-2	T04784	PF1	T08989	Pra-1	T10276	Ems
T00112	c-Ets-1	T00923	Zta	T02100	Zeste	T04811	FOXP1a	T08994	HIF-1alpha-isoform1	T10282	Ota2
T00113	c-Ets-2	T00925	AMT1	T02128	SAP-1b	T04817	LIM1	T09001	BPC1	T10317	IA-1
T00115	c-Ets-1 68	T00937	HBP-1a	T02142	OCA-B	T04819	EmBP-1a	T09018	N-Myc	T10331	NRF-1
T00117	CF1	T00938	HBP-1b	T02216	TFIIA-alpha/beta precursor (major)	T04886	Tel-2b	T09033	TEF-1	T10392	GATA-3
T00120	CF2-II	T00969	POU3F1	T02217	TFIIA-alpha/beta precursor (minor)	T04931	p73alpha	T09051	AhR	T10393	GATA-2
T00128	HOXA4	T01005	MEF2A-isoform1	T02235	PEBP2alphaB1	T04957	EKLF	T09059	SEF2-1B	T10429	PU.1
T00140	c-Myc	T01017	CRE-BP2	T02248	StuAp	T04961	GLI2alpha	T09071	AG	T10459	Alx-3
T00151	CP2a	T01019	Elf-1	T02256	AML1a	T04996	ZBP89	T09089	PIF3	T10462	Prop-1
T00163	CREB	T01027	BAS1	T02288	HFH-1	T04998	Tel-2a	T09093	IPF1	T10473	TEF-5
T00167	ATF-2-xbb4	T01035	lsl-1alpha	T02290	FOXD3	T04999	Tel-2c	T09097	SRY	T10482	AP-2gamma
T00176	CTF-1	T01051	FOXA4a	T02291	Croc	T05021	NERF-1a	T09098	SREBP-2	T10484	TEF-3
T00177	CTF-2	T01053	HNF-3beta	T02294	FOXI1a	T05051	BTEB3	T09102	FOXO4	T10543	Sox5
T00179	CUP2	T01059	MNB1a	T02302	GCM	T05137	CIZ6-1	T09106	RelA-p65	T10573	DREB1A
T00183	DBP	T01072	TEF	T02313	MIBP1	T05181	DSF	T09117	E2F-1	T10588	Snail
T00193	D6l	T01074	Ap	T02330	G/HBF-1	T05553	MYBAS1	T09129	BCL-6	T10638	HY5
T00204	E12	T01078	GBF1	T02361	CREBbeta	T05587	BZI-1	T09156	TGIF-isoform2	T10644	MTF-1
T00208	E74A	T01083	NF-muNR	T02378	USF1	T05682	ERRalpha1	T09158	BZR1	**T10664**	**Gfi1**
T00217	EcR	T01085	abaA	T02419	Sp3	T05705	GATA-1	T09159	PITX2A	T10666	SRY
T00253	En	T01109	TCF-1(P)	T02420	Sox13	T05706	GATA-2	T09162	Pax-3	T10674	MafK
T00262	ER-alpha	T01112	EBF1-L	T02422	HNF-4alpha2	T05707	GATA-3	T09177	MyoD	T10712	DMRT1
T00264	ER-alpha	T01147	SF-1 isoform2	T02429	HNF-4alpha1	T05708	GATA-4	T09178	C/EBPalpha	T10720	GCR1
T00272	Evx	T01152	T3R-alpha1	T02463	GBF1	T05737	PCF3	T09182	Pax-5	T10721	DMRT2
T00295	Ftz	T01154	c-Rel	T02469	AP-2beta	T05743	AB14	T09183	WRKY53	T10723	DMRT3
T00296	FTZ-F1	T01258	MSN4	T02529	PPARgamma1	T05770	DREB1A	T09184	Pax-8	T10725	DMRT7
T00301	GAGA factor	T01265	MAC1	T02636	CBF1	T05834	CBF2	T09190	AGL15	T10727	DMRT4
T00302	GAL4	T01274	ABF2	T02639	ANT	T05835	DRF1.1	T09194	NF-AT1C	T10731	DMRT5
T00303	GAL80	T01275	mat1-Mc	T02654	ERF2	T05837	DRF1.3	T09195	SPL14	T10739	MRP1
T00315	GBF	T01286	ROX1	T02669	EmBP-1a	T05929	SUSIBA2	T09196	HSF2A	T10745	HSFA2
T00329	Glass	T01313	ATF3	T02672	GBF1	T05943	FOXP1d	T09199	STAT5A	T10747	MTF-1
T00330	GLI1	T01333	RXR-gamma	T02690	Dof2	T05975	E2F1	T09218	Mac-1	T10754	ABF1
T00331	GLI3	T01346	Arnt	T02691	Dof3	T05977	PEND	T09225	Ets-1	T10760	HAP1
T00337	GR-alpha	T01350	T3R-beta2	T02772	GCNF	T05982	POTH1	T09226	Lhx2	T10795	C/EBPgamma
T00349	HAP2	T01352	PPARalpha	T02786	RITA-1	T06004	DeltaNp63alpha	T09230	Prep1	T10849	STB5
T00350	HAP3	T01388	C/EBP	T02789	bZIP910	T06029	Sox17	T09243	MafG	T10854	GCN4
T00368	HNF-1alpha-A	T01400	Ets-1 deltaVII	T02790	bZIP911	T06043	AGP1	T09287	MITF-A2	T10881	TRAB1
T00377	HOXA5	T01422	ste11	T02807	OSBZ8	T06137	p73beta	T09304	Smad4	T10928	TGA2
T00383	HSF	T01427	p300	T02809	ROM1	T06168	p63alpha	T09319	IRF-1	T10958	ATHB-2
T00385	HSF1	T01431	c-Maf (long form)	T02810	ROM2	T06341	BEL5	T09323	IRF-1	T10959	PCF1
T00386	HSTF	T01470	Ik-2	T02818	GLN3	T06356	Rmi101p	T09343	SRF	T10960	PCF2
T00395	Hb	T01471	Ik-3	T02825	gaf2	T06401	WRKY38	T09355	Alx-4	T11115	ZIC1
T00401	ICP4	T01476	Abd-B	T02841	FACB	T06429	HIC-1-isoform2	T09356	HOXA3	T11136	DEC2
T00445	KNIRPS	T01477	BR-C Z1	T02846	UAY	T06532	NAC69-1	T09383	GABP-alpha	T11158	HELIOS-B
T00456	Kr	T01478	BR-C Z2	T02878	TCF-4E	T06533	MYB80	T09424	WRKY2	T11164	FOXJ1
T00458	LAC9	T01479	BR-C Z3	T02897	Sox6-Isoform1	T06537	Ci	T09426	Sp3-isoform1	T11166	FOXF1
T00459	C/EBPbeta(LAP)	T01480	BR-C Z4	T02905	LEF-1	T08158	ABZ1	T09427	RAP-1-xbb1	T11180	Gli1
T00480	MAL63	T01481	Pbx1a	T02907	MYB305	T08251	FBI-1	T09431	Sp1	T11200	DEC1
T00487	MATalpha2	T01482	Exd	T02929	MYB340	T08252	NF-AT3	T09441	RBP-Jkappa	T11217	Gzf1
T00488	MATa1	T01484	Cdx-1	T02936	FOXO1	T08279	USF1	T09444	CPRF-3	T11246	ZIC2
T00489	Max-isoform2	T01492	STAT1alpha	T02983	Pax-4a	T08291	GATA-1	T09449	CPRF-2	T11250	Brachyury
T00490	MAZ	T01517	Twi	T02999	OCSBF-1	T08292	GATA-1 isoform 1	T09450	CPRF-1	T11256	GCMb
T00497	MBP-1 (1)	T01527	RORalpha1	T03031	Pax-2.1	T08293	GATA-1	T09462	Egr-1	T11258	GCMa
T00500	MCM1	T01528	RORalpha2	T03178	SQUA	T08298	Kaiso	T09478	TGA1a	T11310	MafA
T00509	MIG1	T01556	SREBP-1a	T03227	CATa	T08300	ER-alpha-L	T09507	Sox-xlbl	T11372	HOXB8
T00529	MZF1B-C	T01590	P (long form)	T03256	HNF-3beta	T08313	USF2a	T09514	HTF4gamma	T11383	HOXD13
T00535	NF-1	T01592	C1 (long form)	T03258	HNF-6beta	T08318	Elf-1	T09531	ATF-4	T11390	Cart-1
T00594	RelA-p65	T01599	LCR-F1	T03388	Meis-1a	T08319	Zec	T09540	c-Krox	T11394	PR-beta
T00625	ZEB (1124 AA)	T01615	Su	T03389	Meis-1b	T08321	p53-isoform1	T09548	IRF-3	T11402	Crx
T00627	NIT2	T01649	HES-1	T03447	LHX3b	T08323	p53	T09561	Roaz	T11425	Chx10
T00642	POU2F1	T01660	PR-alpha	T03481	SKN7	T08340	Egr-2	T09569	Hlf	T11440	FAC1-xbbl
T00644	POU2F1a	T01661	PR A	T03491	MED8	T08348	RXR-alpha	T09571	MYB1	T11453	TAF-1
T00651	POU5F1	T01664	TR2-11	T03500	MOT3	T08358	GATA-4	T09588	E4BP4	T13753	HstB1
T00653	POU5F1 (Oct-5)	T01667	RFX2	T03524	PDR1	T08409	GAMYB	T09608	Kid3	T13760	ABF1
T00669	Ovo-B	T01669	RFX2	T03525	PDR3	T08410	PBF	T09623	ATF6	T13794	TGA1
T00677	Pax-1	T01670	RFX3	T03538	RCS1	T08411	SED	T09629	MYBJS1	T13809	AGL2
T00680	PHO2	T01671	RFX3	T03541	RFX1	T08415	CBT	T09635	AP1	T13810	Dof 4
T00690	PHO4	T01673	RFX1	T03556	RGT1	T08431	PPARalpha	T09649	cel-let-7	T13811	AGL3
T00691	Pit-1A	T01675	Nkx2-5	T03593	Pax-9a	T08441	Sox10	T09701	cel-miR-84	T14002	GKLF
T00696	PR B	T01679	PacC	T03594	Pax-9b	T08445	Elk-1-isoform1	T09706	hsa-let-7a	T14118	ASR-1
T00697	PR B	T01692	T3R-beta1	T03600	SIP4	T08466	c-Jun	T09707	hsa-let-7b	T14187	AIRE-isoform1
T00699	Prd	T01705	HOXA7	T03612	Nk-4	T08475	GR-alpha	T09718	hsa-miR-23a	T14230	WRKY 40
T00709	qs-1F	T01710	HoxA-9	T03707	XBP1	T08482	VDR	T09727	hsa-miR-103	T14231	RP5a
T00710	R	T01735	HOXB7	T03717	ZAP1	T08487	AR	T09729	hsa-miR-107	T14234	WRKY18
T00715	RAP1	T01737	HOXB8	T03718	WRKY1	T08492	LRH-1-xbl1	T09731	dme-miR-2a	T14258	Nkx3-2
T00719	RAR-alpha1	T01755	HOXD9	T03722	ZAP1	T08493	c-Fos	T09732	dme-miR-2b	T14268	MIZF
T00725	REB1	T01757	HOXD10	T03975	SPF1	T08505	COUP-TF1	T09737	dme-miR-7	T14302	C1-Myb
T00731	RME1	T01784	MEF-2A	T03994	ID1	T08520	TBP	T09741	dme-miR-13a	T14317	Myb-15
T00737	SAP-1a	T01786	E12	T04001	ATHB-9	T08528	AR	T09742	dme-miR-13b	T14381	ATHB-1
T00746	SGF-3	T01799	Tal-1	T04096	Smad3	T08544	MOVO-B	T09793	dme-let-7	T14382	ATHB-5
T00751	Sn	T01814	Pax-6 / Pd-5a	T04146	HLTF	T08546	Ovo1a	T09806	hsa-miR-1	T14442	STF1
T00761	SRF	T01823	Pax-2	T04166	FOXD3	T08571	GATA-2	T09810	hsa-miR-124a	T14444	TGA1
T00763	SRF	T01838	Sox4	T04169	FOXJ2 (long isoform)	T08577	ZBRK1	T09812	hsa-miR-130a	T14447	PBF
T00767	Sry-delta	T01841	WT1-del2	T04176	FOXO4	T08580	STAT3	T09819	hsa-miR-125a	T14485	XBP-1
T00769	Sry-beta	T01851	HMGI	T04255	Nkx3-1	T08583	CCA1	T09824	hsa-miR-206	T14491	CBNAC
T00776	SWI5	T01865	Oct-2.3	T04280	FOXP3	T08584	LHY	T09840	hsa-miR-130b	T14517	Zic3
T00788	T-Ag	T01866	Oct-2.4	T04297	Nkx6-1	T08613	ZNF219	T09880	dre-miR-430a	T14521	ZF5
T00789	Tll	T01867	Oct-2.6	T04312	NURR1-isoform 1	T08615	PLZFB	T09892	c-Myb-isoform1	T14543	CBF1
T00798	TBP	T01882	unc-86	T04323	Nkx2-5	T08619	WEREWOLF	T09914	SF-1	T14573	FUS3
T00810	TFE3-L	T01888	POU6F1 (c2)	T04324	DREF	T08621	HAHB-4	T09923	RREB-1	T14681	Spz1
T00812	TFEB-isoform1	T01897	Ct1a	T04336	Nkx2-8	T08624	Sox9	T09942	HNF-3beta	T14827	DEAF-1
T00814	TFE3-S	T01900	PDM-1	T04337	Nkx2-2	T08630	CAR	T09949	FOXC1	T14951	Nex
T00830	TGA1b	T01944	NF-AT1	T04345	TBX5-L	T08667	SZF1-1	T09960	TR4	T14954	OG-2
										T14992	Pitx3

A.1 Problem Definition in Chapter 3

m transcription factors $\{TF^1, TF^2, ..., TF^m\}$ are concerned. Each transcription factor TF^i chemically binds to several binding sites, which are collectively called transcription factor binding sites. Each of them is denoted by $TFBS^i_j$, the j-th transcription factor binding site which chemically binds to TF^i. For example, $TFBS^2_5$ denotes the 5-th transcription factor binding site which chemically binds to TF^2. Since different transcription factors can bind to different number of transcription factor binding sites, N^i denotes the number of the transcription factor binding sites which can chemically bind to TF^i. For example, we have the following transcription factor binding sites $\{TFBS^3_1, TFBS^3_2, ..., TFBS^3_{N^3}\}$ which can chemically bind to TF^3. They are formally defined as follows:

Definition 3.1 TF^i denotes the i-th transcription factor where $1 \leq i \leq m$.

Definition 3.2 $TFBS^i_j$ denotes the j-th transcription factor binding site which chemically binds to TF^i where $1 \leq i \leq m$ and $1 \leq j \leq N^i$.

To discover the binding sequence patterns between m transcription factors and their transcription factor binding sites. Sequence data is introduced. But, before doing so, some basic notations needs to be defined: $AAseq$ denotes a string with the amino acid alphabet, $DNAseq$ denotes a string with the DNA alphabet, and $kmer$ denotes a string with length $= k$. Thus $AAseq^i$ denotes the $AAseq$ of TF^i, whereas $DNAseq^i_j$ denotes the $DNAseq$ of $TFBS^i_j$.

The following sequence dataset is given for each transcription factor : the amino acid sequence of the transcription factor and the DNA sequences of the transcription factor binding sites which can chemically bind to the transcription factor. The dataset for TF^i is thus denoted by $TFdataset^i$: $\{AAseq^i, DNAseq^i_j | \forall j \in \mathbb{N}, j \leq N_i\}$. In total, $\{TFdataset^1, TFdataset^2, ..., TFdataset^m\}$ is given in this problem. To dig out binding sequence patterns from them, we would like to know which kmers are found in the amino acid sequence $AAseq^i$. Thus a function F (Found) is defined:

Definition 3.3 If the kmer A is a substring of the string S, $F(A, S) = 1$. Otherwise, $F(A, S) = 0$.

In each $TFdataset^i$, we would also like to know which kmers are commonly found in the DNA sequences $\{DNAseq^i_j | \forall j \in \mathbb{N}, j \leq N_i\}$. Thus a function CF^i (Commonly Found) is defined:

Definition 3.4 Let $Threshold_{CF} \in [0, 1]$, and $DNAseq^i_j$ be the $DNAseq$ of $TFBS^i_j$. If

$\frac{\sum_{j=1}^{N^i} F(A,DNAseq_j^i)}{N^i} \geq Threshold_{CF}$, $CF^i(A) = 1$. Otherwise, $CF^i(A) = 0$.

In particular, we are interested in the pairs of kmers in which one is found in the amino acid sequences of some transcription factors while the other is commonly found in the DNA sequences of their transcription factor binding sites. Such pairs are then called 'frequently co-occurring' defined as follows:

Definition 3.5 Let $Threshold_{Support} \in \mathbb{N}^+$, $k_1 \in \mathbb{N}^+$, and $k_2 \in \mathbb{N}^+$. A pair of kmers (A-B) is frequently co-occurring if $\frac{\sum_{i=1}^{m} CF^i(A) \times F(B,AAseq^i)}{m} \geq Threshold_{Support}$ where A is a k_1mer and B is a k_2mer.

We assume that the pairs of kmers frequently co-occurring are the binding sequence patterns which we aims to find. Thus the problem definition is summarized as follows: Given $Threshold_{CF} \in [0, 1]$, $Threshold_{Support} \in \mathbb{N}^+$, $k_1 \in \mathbb{N}^+$, $k_2 \in \mathbb{N}^+$, we would like to find all pairs (A-B) frequently co-occurring where A is a k_1mer and B is a k_2mer.

Bibliography

[1] J. L. Moreland, A. Gramada, O. V. Buzko, Q. Zhang, and P. E. Bourne. The Molecular Biology Toolkit (MBT): a modular platform for developing molecular visualization applications. *BMC Bioinformatics*, 6:21, 2005.

[2] Wolfgang Banzhaf, Peter Nordin, Robert E. Keller, and Frank D. Francone. *Genetic Programming - An Introduction; On the Automatic Evolution of Computer Programs and its Applications.* Morgan Kaufmann, San Francisco, CA, USA, January 1998. ISBN 3-920993-58-6. URL `http://www.elsevier.com/wps/find/bookdescription.cws_home/677869/description#description`.

[3] D. J. Galas and A. Schmitz. DNAse footprinting: a simple method for the detection of protein-DNA binding specificity. *Nucleic Acids Res.*, 5(9):3157–3170, September 1987.

[4] M. M. Garner and A. Revzin. A gel electrophoresis method for quantifying the binding of proteins to specific DNA regions: application to components of the escherichia coli lactose operon regulatory system. *Nucleic Acids Res.*, 9(13):3047–3060, July 1981.

[5] A. D. Smith, P. Sumazin, D. Das, and M. Q. Zhang. Mining ChIP-chip data for transcription factor and cofactor binding sites. *Bioinformatics*, Suppl 1(20): i403–i412, 2005.

[6] K. D. MacIsaac and E. Fraenkel. Practical strategies for discovering regulatory dna sequence motifs. *PLoS Comput Biol.*, 2(4):e36, 2006.

[7] X. S. Liu, D. L. Brutlag, and J. S. Liu. An algorithm for finding protein–DNA binding sites with applications to chromatinimmunoprecipitation microarray experiments. *Nat. Biotechnol.*, 20:835–839, 2002.

[8] V. Matys, Olga V. Kel-Margoulis, Ellen Fricke, Ines Liebich, Sigrid Land, A. Barre-Dirrie, Ingmar Reuter, D. Chekmenev, Mathias Krull, Klaus Hornischer, Nico Voss, P. Stegmaier, Birgit Lewicki-Potapov, H. Saxel, Alexander E. Kel, and Edgar Wingender. Transfac and its module transcompel: transcriptional gene regulation in eukaryotes. *Nucleic Acids Research*, 34:108–110, 2006.

[9] Nicolas Hulo, Amos Bairoch, Virginie Bulliard, Lorenzo Cerutti, Beatrice A. Cuche, Edouard de Castro, Corinne Lachaize, Petra S. Langendijk-Genevaux, and Christian J. A. Sigrist. The 20 years of prosite. *Nucl. Acids Res.*, 36(suppl_1): D245–249, January 2008.

[10] Alex Bateman, Lachlan Coin, Richard Durbin, Robert D. Finn, Volker Hollich, Sam GrifRths-Jones, Ajay Khanna, Mhairi Marshall, Simon Moxon, Erik L. L.

Sonnhammer, David J. Studholme, Corin Yeats, and Sean R. Eddy. The pfam protein families database. *Nucleic Acids Res*, 32:D138–141, 2004.

[11] Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The protein data bank. *Nucl. Acids Res.*, 28(1):235–242, January 2000. ISSN 0305-1048. doi: 10.1093/nar/28.1.235. URL `http://dx.doi.org/10.1093/nar/28.1.235`.

[12] R.D. Fleischmann, M.D. Adams, O. White, R.A. Clayton, E.F. Kirkness, A.R. Kerlavage, C.J. Bult, J.F. Tomb, B.A. Dougherty, and J.M. Merrick. Whole-genome random sequencing and assembly of Haemophilus influenzae Rd. *Science*, 269:496–512, Jul 1995.

[13] A. Goffeau, B.G. Barrell, H. Bussey, R.W. Davis, B. Dujon, H. Feldmann, F. Galibert, J.D. Hoheisel, C. Jacq, M. Johnston, E.J. Louis, H.W. Mewes, Y. Murakami, P. Philippsen, H. Tettelin, and S.G. Oliver. Life with 6000 genes. *Science*, 274: 563–567, Oct 1996.

[14] Arabidopsis Genome Initiative. Analysis of the genome sequence of the flowering plant Arabidopsis thaliana. *Nature*, 408:796–815, Dec 2000.

[15] S.G. Gregory, K.F. Barlow, K.E. McLay, R. Kaul, D. Swarbreck, A. Dunham, C.E. Scott, K.L. Howe, K. Woodfine, and et al. The DNA sequence and biological annotation of human chromosome 1. *Nature*, 441:315–321, May 2006.

[16] D. GuhaThakurta. Computational identification of transcriptional regulatory elements in DNA sequence. *Nucleic Acids Res.*, 34:3585–3598, 2006.

[17] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, October 1990. ISSN 0022-2836. doi: 10.1006/jmbi.1990.9999. URL `http://dx.doi.org/10.1006/jmbi.1990.9999`.

[18] L. J. McGuffin, K. Bryson, and D. T. Jones. The psipred protein structure prediction server. *Bioinformatics (Oxford, England)*, 16(4):404–405, April 2000. ISSN 1367-4803. doi: 10.1093/bioinformatics/16.4.404. URL `http://dx.doi.org/10.1093/bioinformatics/16.4.404`.

[19] P. D'Haeseleer, S. Liang, and R. Somogyi. Genetic network inference: from co-expression clustering to reverse engineering. *Bioinformatics (Oxford, England)*, 16(8):707–726, August 2000. ISSN 1367-4803. doi: http://dx.doi.org/10.1093/bioinformatics/16.8.707. URL `http://dx.doi.org/10.1093/bioinformatics/16.8.707`.

[20] Thomas Abeel, Yves Van de Peer, and Yvan Saeys. Toward a gold standard for promoter prediction evaluation. *Bioinformatics*, 25(12):i313–i320, 2009. ISSN 1367-4803. doi: http://dx.doi.org/10.1093/bioinformatics/btp191.

[21] Fredrik Ronquist and John P. Huelsenbeck. Mrbayes 3: Bayesian phylogenetic inference under mixed models. *Bioinformatics*, 19(12):1572–1574, August 2003. ISSN 1367-4803. doi: 10.1093/bioinformatics/btg180. URL `http://dx.doi.org/10.1093/bioinformatics/btg180`.

[22] Yong Liang, Kwong-Sak Leung, and Tony Shu Kam Mok. Evolutionary drug scheduling models with different toxicity metabolism in cancer chemotherapy. *Appl. Soft Comput.*, 8(1):140–149, 2008. ISSN 1568-4946. doi: http://dx.doi.org/10.1016/j.asoc.2006.12.002.

[23] Lila Kari and Grzegorz Rozenberg. The many facets of natural computing. *Commun. ACM*, 51(10):72–83, 2008. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/1400181.1400200.

[24] John H. Holland. *Adaptation in natural and artificial systems.* MIT Press, Cambridge, MA, USA, 1992. ISBN 0-262-58111-6.

[25] Jouni Lampinen and Ivan Zelinka. Mechanical engineering design optimization by differential evolution. *New ideas in optimization*, pages 127–146, 1999.

[26] Yahya Rahmat-Samii and Eric Michielssen, editors. *Electromagnetic Optimization by Genetic Algorithms.* John Wiley & Sons, Inc., New York, NY, USA, 1999. ISBN 0471295450.

[27] Ilaria Bertini, Matteo De Felice, Fabio Moretti, and Stefano Pizzuti. Start-up optimisation of a combined cycle power plant with multiobjective evolutionary algorithms. In *EvoApplications (2)*, pages 151–160, 2010.

[28] Fiacc Larkin and Conor Ryan. Modesty is the best policy: Automatic discovery of viable forecasting goals in financial data. In *EvoApplications (2)*, pages 202–211, 2010.

[29] Philippe Esling, Grégoire Carpentier, and Carlos Agon. Dynamic musical orchestration using genetic algorithms and a spectro-temporal description of musical instruments. In *EvoApplications (2)*, pages 371–380, 2010.

[30] Marcus Furuholmen, Kyrre Glette, Mats Høvin, and Jim Torresen. Evolutionary approaches to the three-dimensional multi-pipe routing problem: A comparative study using direct encodings. In *EvoCOP*, pages 71–82, 2010.

[31] W. F. Sacco, N. Henderson, A. C. Rios-Coelho, M. M. Ali, and C. M. N. A. Pereira. Differential evolution algorithms applied to nuclear reactor core design. *Annals of Nuclear Energy*, June 2009. ISSN 03064549. doi: 10.1016/j.anucene.2009.05.007. URL `http://dx.doi.org/10.1016/j.anucene.2009.05.007`.

[32] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989. ISBN 0201157675.

[33] Qing Zhou and Jun S. Liu. Extracting sequence features to predict protein-dna interactions: a comparative study. *Nucl. Acids Res.*, 36(12):4137–4148, July 2008. doi: 10.1093/nar/gkn361. URL `http://dx.doi.org/10.1093/nar/gkn361`.

[34] Shandar Ahmad, Michael M. Gromiha, and Akinori Sarai. Analysis and prediction of dna-binding proteins and their binding residues based on composition, sequence and structural information. *Bioinformatics*, 20(4):477–486, March 2004. ISSN 1367-4803. doi: 10.1093/bioinformatics/btg432. URL `http://dx.doi.org/10.1093/bioinformatics/btg432`.

[35] S. Ahmad, O. Keskin, A. Sarai, and R. Nussinov. Protein-DNA interactions: structural, thermodynamic and clustering patterns of conserved residues in DNA-binding proteins. *Nucleic Acids Res.*, 36:5922–5932, Oct 2008.

[36] Tho Hoan Pham, José Carlos Clemente, Kenji Satou, and Tu Bao Ho. Computational discovery of transcriptional regulatory rules. *Bioinformatics*, 21(2):101–107, 2005. ISSN 1367-4803. doi: http://dx.doi.org/10.1093/bioinformatics/bti1117.

[37] Yanay Ofran, Venkatesh Mysore, and Burkhard Rost. Prediction of dna-binding residues from sequence. *Bioinformatics*, 23(13):i347–353, July 2007. doi: 10.1093/bioinformatics/btm174. URL `http://dx.doi.org/10.1093/bioinformatics/btm174`.

[38] A. Sarai and H. Kono. Protein-dna recognition patterns and predictions. *Annu Rev Biophys Biomol Struct*, 34:379–398, 2005. ISSN 1056-8700. URL `http://dx.doi.org/10.1146/annurev.biophys.34.040204.144537`.

[39] N. M. Luscombe and J. M. Thornton. Protein-dna interactions: amino acid conservation and the effects of mutations on binding specificity. *J Mol Biol*, 320(5): 991–1009, July 2002. ISSN 0022-2836. URL `http://view.ncbi.nlm.nih.gov/pubmed/12126620`.

[40] N. M. Luscombe, S. E. Austin, H. M. Berman, and J. M. Thornton. An overview of the structures of protein-dna complexes. *Genome Biol.*, 1(1):REVIEWS001, June 2000.

[41] A. E. Kel, E. Goessling, I. Reuter, E. Cheremushkin, O. V. Kel-Margoulis, and E. Wingender. MATCH: a tool for searching transcription factor binding sites in DNA sequences. *Nucleic Acids Res.*, 31(13):3576–3579, 2003.

[42] Gary D. Stormo. Computer methods for analyzing sequence recognition of nucleic acids. *Annu. Rev. BioChem.*, 17:241–263, 1988.

[43] S. T. Jensen, X. S. Liu, Q. Zhou, and J. S. Liu. Computational discovery of gene regulatory binding motifs: a bayesian perspective. *Statistical Science*, 19(1): 188–204, 2004.

[44] M. Tompa, N. Li, T. L. Bailey, G. M. Church, B. De Moor, E. Eskin, A. V. Favorov, M. C. Frith, Y. Fu, W. J. Kent, V. J. Makeev, A. A. Mironov, W. S. Noble, G. Pavesi, G. Pesole, M. Regnier, N. Simonis, S. Sinha, G. Thijs, J. van Helden, M. Vandenbogaert, Z. Weng, C. Workman, C. Ye, , and Z. Zhu. Assessing computational tools for the discovery of transcription factor binding sites. *Nature Biotechnology*, 23(1):137–144, 2005.

[45] Geir Kjetil Sandve, Osman Abul, Vegard Walseng, and Finn Drablos. Improved benchmarks for computational motif discovery. *BMC Bioinformatics*, 8(1):193, 2007.

[46] S. Jones, P. van Heyningen, H. M. Berman, and J. M. Thornton. Protein-dna interactions: a structural analysis. *Journal of Molecular Biology*, 287(5):877–896, April 1999. ISSN 00222836. doi: 10.1006/jmbi.1999.2659. URL `http://dx.doi.org/10.1006/jmbi.1999.2659`.

[47] N. M. Luscombe, R. A. Laskowski, and J. M. Thornton. Amino acid-base interactions: a three-dimensional analysis of protein-dna interactions at an atomic level. *Nucleic Acids Res*, 29(13):2860–2874, July 2001. ISSN 1362-4962. URL `http://nar.oxfordjournals.org/cgi/content/abstract/29/13/2860`.

[48] Sri S. Krishna, Indraneel Majumdar, and Nick V. Grishin. Structural classification of zinc fingers: survey and summary. *Nucleic acids research*, 31(2):532–550, January 2003. ISSN 1362-4962. URL `http://view.ncbi.nlm.nih.gov/pubmed/12527760`.

[49] S. Jones, H. P. Shanahan, H. M. Berman, and J. M. Thornton. Using electrostatic potentials to predict dna-binding sites on dna-binding proteins. *Nucleic Acids Res*, 31(24):7189–7198, December 2003. ISSN 1362-4962. doi: 10.1093/nar/gkg922. URL `http://dx.doi.org/10.1093/nar/gkg922`.

[50] Y. Mandel-Gutfreund, O. Schueler, and H. Margalit. Comprehensive analysis of hydrogen bonds in regulatory protein dna-complexes: in search of common principles. *J Mol Biol*, 253(2):370–382, October 1995. ISSN 0022-2836. doi: 10.1006/jmbi.1995.0559. URL `http://dx.doi.org/10.1006/jmbi.1995.0559`.

[51] Y. Mandel-Gutfreund and H. Margalit. Quantitative parameters for amino acid-base interaction: implications for prediction of protein-dna binding sites. *Nucleic Acids Res*, 26(10):2306–2312, May 1998. ISSN 0305-1048. doi: 10.1093/nar/26.10.2306. URL `http://dx.doi.org/10.1093/nar/26.10.2306`.

[52] V. Matys, O.V. Kel-Margoulis, E. Fricke, I. Liebich, S. Land, A. Barre-Dirrie, I. Reuter, D. Chekmenev, M. Krull, K. Hornischer, N. Voss, P. Stegmaier, B. Lewicki-Potapov, H. Saxel, A.E. Kel, and E. Wingender. TRANSFAC and its module TRANSCompel: transcriptional gene regulation in eukaryotes. *Nucleic Acids Res.*, 34:D108–110, Jan 2006.

[53] E. Wingender, P. Dietze, H. Karas, and R. Knppel. TRANSFAC: a database on transcription factors and their DNA binding sites. *Nucleic Acids Res.*, 24:238–241, Jan 1996.

[54] E. Wingender, X. Chen, E. Fricke, R. Geffers, R. Hehl, I. Liebich, M. Krull, V. Matys, H. Michael, R. Ohnhuser, M. Prss, F. Schacherer, S. Thiele, and S. Urbach. The TRANSFAC system on gene expression regulation. *Nucleic Acids Res.*, 29:281–283, Jan 2001.

[55] G. Robertson, M. Bilenky, K. Lin, A. He, W. Yuen, M. Dagpinar, R. Varhol, K. Teague, O.L. Griffith, X. Zhang, Y. Pan, M. Hassel, M.C. Sleumer, W. Pan, E.D. Pleasance, M. Chuang, H. Hao, Y.Y. Li, N. Robertson, C. Fjell, B. Li, S.B. Montgomery, T. Astakhova, J. Zhou, J. Sander, A.S. Siddiqui, and S.J. Jones. cisRED: a database system for genome-scale computational discovery of regulatory elements. *Nucleic Acids Res.*, 34:68–73, Jan 2006.

[56] S.B. Montgomery, O.L. Griffith, M.C. Sleumer, C.M. Bergman, M. Bilenky, E.D. Pleasance, Y. Prychyna, X. Zhang, and S.J. Jones. ORegAnno: an open access database and curation system for literature-derived promoters, transcription factor binding sites and regulatory variation. *Bioinformatics*, 22:637–640, Mar 2006.

[57] Kenneth A. De Jong. *Evolutionary Computation. A Unified Approach.* MIT Press, Cambridge, MA, USA, 2006. ISBN 0-262-04194-4.

[58] N. Krasnogor, W.E. Hart, J. Smith, and D. Pelta. Protein structure prediction with evolutionary algorithms. In *International Genetic and Evolutionary Computation Conference (GECCO99)*, pages 1569–1601. Morgan Kaufmann, 1999. URL `http://www.cs.nott.ac.uk/~nxk/PAPERS/gecco99.pdf`.

[59] Rainer Storn and Kenneth Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, December 1997. ISSN 0925-5001 (Print) 1573-2916 (Online). doi: 10.1023/A:1008202821328. URL `http://www.springerlink.com/content/x555692233083677/`.

[60] David E. Goldberg and Jon Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic algorithms and their application*, pages 41–49, Hillsdale, NJ, USA, 1987. L. Erlbaum Associates Inc. ISBN 0-8058-0158-8.

[61] Zyed Bouzarkouna, Anne Auger, and Didier Yu Ding. Investigating the local-meta-model cma-es for large population sizes. In *EvoApplications (1)*, pages 402–411, 2010.

[62] Darrell Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4(2):65–85, June 1994. doi: 10.1007/BF00175354. URL `http://dx.doi.org/10.1007/BF00175354`.

[63] R. Storn and K. Price. Differential evolution- a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, International Computer Science Institute, 1995. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.1.9696`.

[64] H. Bersini, M. Dorigo, S. Langerman, G. Seront, and L. Gambardella. Results of the first international contest on evolutionaryoptimisation (1st ICEO). In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 611–615, Nagoya, Japan, May 1996. doi: 10.1109/ICEC.1996.542670.

[65] K. V. Price. Differential evolution vs. the functions of the 2nd ICEO. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 153–157, Indianapolis, IN, USA, April 1997. doi: 10.1109/ICEC.1997.592287.

[66] Vitaliy Feoktistov. *Differential Evolution: In Search of Solutions (Springer Optimization and Its Applications)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387368957.

[67] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies - a comprehensive introduction. *Natural Computing*, 1(1):3–52, March 2002. ISSN 1567-7818. doi: 10.1023/A:1015059928466. URL `http://dx.doi.org/10.1023/A:1015059928466`.

[68] M. Dorigo and L. M. Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *Evolutionary Computation, IEEE Transactions on*, 1(1):53–66, 1997. doi: 10.1109/4235.585892. URL `http://dx.doi.org/10.1109/4235.585892`.

[69] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. *Swarm Intelligence*, 1(1):33–57, June 2007. ISSN 1935-3812. doi: 10.1007/s11721-007-0002-0. URL `http://dx.doi.org/10.1007/s11721-007-0002-0`.

[70] Dervis Karaboga, Bahriye Akay, and Celal Ozturk. Artificial bee colony (abc) optimization algorithm for training feed-forward neural networks. In Vicen? Torra, Yasuo Narukawa, and Yuji Yoshida, editors, *MDAI*, volume 4617 of *Lecture Notes in Computer Science*, pages 318–329. Springer, 2007. ISBN 978-3-540-73728-5. URL `http://dblp.uni-trier.de/db/conf/mdai/mdai2007.html#KarabogaA007`.

[71] Steffen Iredi, Daniel Merkle, and Martin Middendorf. Bi-criterion optimization with multi colony ant algorithms. In *in Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001), no. 1993 in LNCS*, pages 359–372. Springer, 2000.

[72] Ruud Schoonderwoerd, Janet L. Bruten, Owen E. Holland, and Leon J. M. Rothkrantz. Ant-based load balancing in telecommunications networks. *Adapt. Behav.*, 5(2):169–207, 1996. ISSN 1059-7123. doi: http://dx.doi.org/10.1177/105971239700500203.

[73] Alena Shmygelska, Rosalía Aguirre Hernández, and Holger H. Hoos. An ant colony optimization algorithm for the 2d hp protein folding problem. In *ANTS '02: Proceedings of the Third International Workshop on Ant Algorithms*, pages 40–53, London, UK, 2002. Springer-Verlag. ISBN 3-540-44146-8.

[74] Del, G. K. Venayagamoorthy, S. Mohagheghi, J. C. Hernandez, and R. G. Harley. Particle swarm optimization: Basic concepts, variants and applications in power systems. *Evolutionary Computation, IEEE Transactions on*, 12(2):171–195, 2008. doi: 10.1109/TEVC.2007.896686. URL `http://dx.doi.org/10.1109/TEVC.2007.896686`.

[75] Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. Algorithms for association rule mining — a general survey and comparison. *SIGKDD Explor. Newsl.*, 2 (1):58–64, 2000. ISSN 1931-0145. doi: http://doi.acm.org/10.1145/360402.360421.

[76] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. *SIGMOD '93: Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216, 1993. doi: http://doi.acm.org/10.1145/170035.170072.

[77] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc. ISBN 1-55860-153-8.

[78] Ashoka Savasere, Edward Omiecinski, and Shamkant B. Navathe. An efficient algorithm for mining association rules in large databases. In *VLDB '95: Proceedings of the 21th International Conference on Very Large Data Bases*, pages 432–444, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1-55860-379-4.

[79] Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. An effective hash-based algorithm for mining association rules. In *SIGMOD '95: Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 175–186, New York, NY, USA, 1995. ACM. ISBN 0-89791-731-6. doi: http://doi.acm.org/10.1145/223784.223813.

[80] Hannu Toivonen. Sampling large databases for association rules. In *VLDB '96: Proceedings of the 22th International Conference on Very Large Data Bases*, pages 134–145, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc. ISBN 1-55860-382-4.

[81] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 1–12, New York, NY, USA, 2000. ACM. ISBN 1-58113-217-4. doi: http://doi.acm.org/10.1145/342009.335372.

[82] CSC5120. Department of computer science and engineering. *The Chinese University of Hong Kong*, 2008. URL `http://appsrv.cse.cuhk.edu.hk/~unprog/csc5120/Lectures/4fptree.pdf`.

[83] Kenneth O. May. A set of independent necessary and sufficient conditions for simple majority decision. *Econometrica*, 20(4):680–684, 1952. ISSN 00129682. doi: 10.2307/1907651. URL `http://dx.doi.org/10.2307/1907651`.

[84] Liqiang Geng and Howard J. Hamilton. Interestingness measures for data mining: A survey. *ACM Comput. Surv.*, 38(3):9, 2006. ISSN 0360-0300. doi: http://doi.acm.org/10.1145/1132960.1132963.

[85] Joy Paul Guilford. *Psychometric Methods*. McGraw-Hill, New York, 1936.

[86] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. Dynamic itemset counting and implication rules for market basket data. *SIGMOD Rec.*, 26(2):255–264, 1997. ISSN 0163-5808. doi: http://doi.acm.org/10.1145/253262.253325.

[87] Derek Wilson, Varodom Charoensawan, Sarah K. Kummerfeld, and Sarah A. Teichmann. DBD taxonomically broad transcription factor predictions: new content and functionality. *Nucl. Acids Res.*, 36:D88–92, 2008. doi: 10.1093/nar/gkm964.

[88] P. L. Privalov and S. J. Gill. Stability of protein structure and hydrophobic interaction. *Adv. Protein Chem.*, 39:191–234, 1988.

[89] J. Moore and E. Devaney. Cloning and characterization of two nuclear receptors from the filarial nematode Brugia pahangi. *Biochem. J.*, 344 Pt 1:245–252, Nov 1999.

[90] M. M. Brent, R. Anand, and R. Marmorstein. Structural basis for DNA recognition by FoxO1 and its regulation by posttranslational modification. *Structure*, 16:1407–1416, Sep 2008.

[91] D. L. Bates, Y. Chen, G. Kim, L. Guo, and L. Chen. Crystal structures of multiple GATA zinc fingers bound to DNA reveal new insights into DNA recognition and self-association by GATA. *J. Mol. Biol.*, 381:1292–1306, Sep 2008.

[92] V. Chandra, P. Huang, Y. Hamuro, S. Raghuram, Y. Wang, T. P. Burris, and F. Rastinejad. Structure of the intact PPAR-gamma-RXR-alpha nuclear receptor complex on DNA. *Nature*, pages 350–356, Oct 2008.

[93] E. P. Lamber, L. Vanhille, L. C. Textor, G. S. Kachalova, M. H. Sieweke, and M. Wilmanns. Regulation of the transcription factor Ets-1 by DNA-mediated homo-dimerization. *EMBO J.*, 27:2006–2017, Jul 2008.

[94] Carl O. Pabo and Robert T. Sauer. Transcription factors: Structural families and principles of dna recognition. *Annual Review of Biochemistry*, 61:1053–1095, 1992.

[95] T. E. Ellenberger, C. J. Brandl, K. Struhl, and S. C. Harrison. The GCN4 basic region leucine zipper binds DNA as a dimer of uninterrupted alpha helices: crystal structure of the protein-DNA complex. *Cell*, 71:1223–1237, Dec 1992.

[96] Y. Sibe'ril, P. Doireau, and P. Gantet. Plant bZIP G-box binding factors. Modular structure and activation mechanisms. *Eur. J. Biochem.*, 268:5655–5666, Nov 2001.

[97] Ling Qing, Wu Gang, Yang Zaiyue, and Wang Qiuping. Crowding clustering genetic algorithm for multimodal function optimization. *Appl. Soft Comput.*, 8(1): 88–95, 2008. ISSN 1568-4946. doi: http://dx.doi.org/10.1016/j.asoc.2006.10.014.

[98] Kenneth Alan De Jong. *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, University of Michigan, Ann Arbor, MI, USA, 1975.

[99] Jian Ping Li, Marton E. Balazs, Geoffrey T. Parks, and P. John Clarkson. A species conserving genetic algorithm for multimodal function optimization. *Evol. Comput.*, 10(3):207–234, 2002. ISSN 1063-6560. doi: http://dx.doi.org/10.1162/106365602760234081.

[100] R. Manner, Samir W. Mahfoud, and Samir W. Mahfoud. Crowding and preselection revisited. In *Parallel Problem Solving From Nature*, pages 27–36. North-Holland, 1992.

[101] Kwong Sak Leung and Yong Liang. Adaptive elitist-population based genetic algorithm for multimodal function optimization. In *In GECCO 2003, LNCS 2723*, pages 1160–1171. Springer-Verlag, 2003.

[102] R. Thomsen. Multimodal optimization using crowding-based differential evolution. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, pages 1382–1389, June 2004. doi: 10.1109/CEC.2004.1331058.

[103] A. Petrowski. A clearing procedure as a niching method for genetic algorithms. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 798–803, Nagoya, Japan, May 1996. doi: 10.1109/ICEC.1996.542703.

[104] David Beasley, David R. Bull, and Ralph R. Martin. A sequential niche technique for multimodal function optimization. *Evol. Comput.*, 1(2):101–125, 1993. ISSN 1063-6560. doi: http://dx.doi.org/10.1162/evco.1993.1.2.101.

[105] Mourad Bessaou, Alain Pétrowski, and Patrick Siarry. Island model cooperating with speciation for multimodal optimization. In *PPSN VI: Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, pages 437–446, London, UK, 2000. Springer-Verlag. ISBN 3-540-41056-2.

[106] Thomas Weise. *Global Optimization Algorithms Theory and Application.* Thomas Weise, july 16, 2007 edition, July 2007. URL `http://www.it-weise.de/projects/book.pdf`. Online available at http://www.it-weise.de/projects/book.pdf.

[107] Gulshan Singh and Dr. Kalyanmoy Deb. Comparison of multi-modal optimization algorithms based on evolutionary algorithms. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1305–1312,

New York, NY, USA, 2006. ACM. ISBN 1-59593-186-4. doi: http://doi.acm.org/10.1145/1143997.1144200.

[108] Kalyanmoy Deb and David E. Goldberg. An investigation of niche and species formation in genetic function optimization. In *Proceedings of the third international conference on Genetic algorithms*, pages 42–50, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc. ISBN 1-55860-006-3.

[109] Zbigniew Michalewicz. *Genetic algorithms + data structures = evolution programs (3rd ed.)*. Springer-Verlag, London, UK, 1996. ISBN 3-540-60676-9.

[110] Yun-Wei Shang and Yu-Huang Qiu. A note on the extended rosenbrock function. *Evol. Comput.*, 14(1):119–126, 2006. ISSN 1063-6560. doi: http://dx.doi.org/10.1162/106365606776022733.

[111] Rodica Ioana Lung and D. Dumitrescu. A new evolutionary model for detecting multiple optima. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1296–1303, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-697-4. doi: http://doi.acm.org/10.1145/1276958.1277204.

[112] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real parameter optimization. Technical report, Nanyang Technological University, 2005.

[113] A. Rogers and K. Pingali. Process decomposition through locality of reference. *SIGPLAN Not.*, 24(7):69–80, 1989. ISSN 0362-1340. doi: http://doi.acm.org/10.1145/74818.74824.

[114] Peter J. Denning. The locality principle. *Commun. ACM*, 48(7):19–24, 2005. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/1070838.1070856.

[115] Xiaodong Li. Efficient differential evolution using speciation for multimodal function optimization. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 873–880, New York, NY, USA, 2005. ACM. ISBN 1-59593-010-8. doi: http://doi.acm.org/10.1145/1068009.1068156.

[116] Ka-Chun Wong, Kwong-Sak Leung, and Man-Hon Wong. An evolutionary algorithm with species-specific explosion for multimodal optimization. In *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 923–930, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-325-9. doi: http://doi.acm.org/10.1145/1569901.1570027.

[117] Rodica I. Lung, Camelia Chira, and D. Dumitrescu. An agent-based collaborative evolutionary model for multimodal optimization. In *GECCO '08: Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*, pages 1969–1976, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-131-6. doi: http://doi.acm.org/10.1145/1388969.1389008.

[118] R. J. White. *Gene Transcription: Mechanisms and Control.* Wiley-Blackwell, 2001. ISBN 978-0632048885.

[119] A. M. McGuire, P. De Wulf, G. M. Church, and E. C. Lin. A weight matrix for binding recognition by the redox-response regulator ArcA-P of Escherichia coli. *Mol. Microbiol.*, 32:219–221, Apr 1999.

[120] B. B. Tuch, D. J. Galgoczy, A. D. Hernday, H. Li, and A. D. Johnson. The evolution of combinatorial gene regulation in fungi. *PLoS Biol.*, 6:e38, Feb 2008.

[121] V. Matys, O. V. Kel-Margoulis, E. Fricke, I. Liebich, S. Land, A. Barre-Dirrie, I. Reuter, D. Chekmenev, M. Krull, K. Hornischer, N. Voss, P. Stegmaier, B. Lewicki-Potapov, H. Saxel, A. E. Kel, and E. Wingender. TRANSFAC and its module TRANSCompel: transcriptional gene regulation in eukaryotes. *Nucleic Acids Res.*, 34:D108–110, Jan 2006.

[122] William M. Spears and Vic Anand. A study of crossover operators in genetic programming. In *ISMIS '91: Proceedings of the 6th International Symposium on Methodologies for Intelligent Systems*, pages 409–418, London, UK, 1991. Springer-Verlag. ISBN 3-540-54563-8.

[123] Riccardo Poli, William B. Langdon, and Nicholas F. Mcphee. *A Field Guide to Genetic Programming.* Lulu Enterprises, UK Ltd, March 2008. ISBN 1409200736. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/1409200736`.

[124] Sean Luke and Lee Spector. A revised comparison of crossover and mutation in genetic programming. In *Proceedings of the Third Annual Genetic Programming Conference (GP98)*, pages 208–213. Morgan Kaufmann, 1998.

[125] David R. White and Simon Poulding. A rigorous evaluation of crossover and mutation in genetic programming. In *EuroGP '09: Proceedings of the 12th European Conference on Genetic Programming*, pages 220–231, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-01180-1. doi: http://dx.doi.org/10.1007/978-3-642-01181-8_19.

[126] Man Leung Wong and Kwong Sak Leung. *Data Mining Using Grammar-Based Genetic Programming and Applications.* Kluwer Academic Publishers, Norwell, MA, USA, 2000. ISBN 079237746X.

[127] Experiments On The, Riccardo Poli, and Stefano Cagnoni. Genetic programming with user-driven selection: Experiments on the evolution of algorithms for image enhancement. In *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 269–277. Morgan Kaufmann, 1997.

[128] Christopher Neely, Paul Weller, Robert Dittmar, Chris Neely, Paul Weller, and Rob Dittmar. Is technical analysis in the foreign exchange market profitable? a genetic programming approach. *Journal of Financial and Quantitative Analysis*, 32:405–426, 1997.

[129] Larry Gritz and James K. Hahn. Genetic programming evolution of controllers for 3-d character animation. In *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 139–146. Morgan Kaufmann, 1997.

[130] Christian B. Anfinsen. Principles that Govern the Folding of Protein Chains. *Science*, 181(4096):223–230, 1973. doi: 10.1126/science.181.4096.223. URL `http://www.sciencemag.org`.

[131] Cyrus Levinthal. Are there pathways for protein folding? *J. Chem. Phys.*, 65: 44–45, 1968.

[132] Hao Li, Robert Helling, Chao Tang, and Ned Wingreen. Emergence of Preferred Structures in a Simple Model of Protein Folding. *Science*, 273(5275):666–669, 1996. doi: 10.1126/science.273.5275.666. URL `http://www.sciencemag.org/cgi/content/abstract/273/5275/666`.

[133] Srinivas Aluru. *Handbook of Computational Molecular Biology (Chapman & All/Crc Computer and Information Science Series)*. Chapman & Hall/CRC, 2005. ISBN 1584884061.

[134] David Baker and Andrej Sali. Protein Structure Prediction and Structural Genomics. *Science*, 294(5540):93–96, 2001. doi: 10.1126/science.1065659. URL `http://www.sciencemag.org/cgi/content/abstract/sci;294/5540/93`.

[135] Heitor Silverio Lopes. Evolutionary algorithms for the protein folding problem: A review and current trends. *Computational Intelligence in Biomedicine and Bioinformatics*, pages 297–315, 2008. doi: 10.1007/978-3-540-70778-3_12. URL `http://www.springerlink.com/content/e810504k11t13107`.

[136] D. Baker. A surprising simplicity to protein folding. *Nature*, 405(6782):39–42, May 2000. ISSN 0028-0836. doi: 10.1038/35011000. URL `http://dx.doi.org/10.1038/35011000`.

[137] Y. Duan and P. A. Kollman. Computational protein folding: from lattice to all-atom. *IBM Syst. J.*, 40(2):297–309, 2001. ISSN 0018-8670.

[138] R. Santana, P. Larranaga, and J. A. Lozano. Protein folding in simplified models with estimation of distribution algorithms. *IEEE Transactions on Evolutionary Computation*, 12(4):418–438, August 2008. ISSN 1089-778X. doi: 10.1109/TEVC.2007.906095.

[139] V. Cutello, G. Nicosia, M. Pavone, and J. Timmis. An immune algorithm for protein structure prediction on lattice models. *IEEE Transactions on Evolutionary Computation*, 11(1):101–117, February 2007. ISSN 1089-778X. doi: 10.1109/TEVC.2006.880328.

[140] Ron Unger and John Moult. Genetic algorithm for 3d protein folding simulations. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 581–588, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc. ISBN 1-55860-299-2.

[141] Arnold L. Patton, William F. Punch, III, and Erik D. Goodman. A standard ga approach to native protein conformation prediction. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 574–581, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1-55860-370-0.

[142] Bonnie Berger and Tom Leighton. Protein folding in the hydrophobic-hydrophilic (hp) is np-complete. In *RECOMB '98: Proceedings of the second annual international conference on Computational molecular biology*, pages 30–39, New York, NY, USA, 1998. ACM. ISBN 0-89791-976-9. doi: http://doi.acm.org/10.1145/279069.279080.

[143] N. Krasnogor, B. Blackburnem, J.D. Hirst, and E.K. Burke. Multimeme algorithms for protein structure prediction. In *7th International Conference Parallel Problem Solving from Nature*, volume 2439 of *Springer Lecture Notes in Computer Science*, pages 769–778, Granada, Spain, September 2002. PPSN, Springer Berlin / Heidelberg. URL `http://www.cs.nott.ac.uk/~nxk/PAPERS/ppsn2002.pdf`. ISBN 3-540-44139-5.

[144] Graham A. Cox, Thomas V. Mortimer-Jones, Robert P. Taylor, and Roy L. Johnston. Development and optimisation of a novel genetic algorithm for studying model protein folding. *Theoretical Chemistry Accounts: Theory, Computation, and Modeling*, 112(3):163–178, 2004. ISSN 1432-881X. doi: 10.1007/s00214-004-0601-4.

[145] T. Hoque, M. Chetty, and L. S. Dooley. A guided genetic algorithm for protein folding prediction using 3d hydrophobic-hydrophilic model. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 2339–2346, Vancouver, BC,, 2006. doi: 10.1109/CEC.2006.1688597.

[146] Alena Shmygelska and Holger Hoos. An ant colony optimisation algorithm for the 2d and 3d hydrophobic polar protein folding problem. *BMC Bioinformatics*, 6 (1):30, 2005. ISSN 1471-2105. doi: 10.1186/1471-2105-6-30. URL `http://www.biomedcentral.com/1471-2105/6/30`.

[147] R. Bitello and H. S. Lopes. A differential evolution approach for protein folding. In *Computational Intelligence and Bioinformatics and Computational Biology, 2006. CIBCB '06. 2006 IEEE Symposium on*, pages 1–5, Toronto, Ont.,, September 2006. doi: 10.1109/CIBCB.2006.331016.

[148] T. R. Jahn and S. E. Radford. Folding versus aggregation: polypeptide conformations on competing pathways. *Arch. Biochem. Biophys.*, 469:100–117, Jan 2008.

[149] R. Unger and J. Moult. Genetic algorithms for protein folding simulations. *J. Mol. Biol.*, 231:75–81, May 1993.

[150] S. D. Flores and J. Smith. Study of fitness landscapes for the HP model of protein structure prediction. In *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, volume 4, pages 2338–2345, December 2003. doi: 10.1109/CEC.2003.1299380.

[151] L. Holm and C. Sander. Protein structure comparison by alignment of distance matrices. *J. Mol. Biol.*, 233:123–138, Sep 1993.

[152] Carlos Cotta. Protein structure prediction using evolutionary algorithms hybridized with backtracking. In *IWANN '03: Proceedings of the 7th International Work-Conference on Artificial and Natural Neural Networks*, pages 321–328, Berlin, Heidelberg, 2003. Springer-Verlag. ISBN 978-3-540-40211-4. doi: http://dx.doi.org/10.1007/3-540-44869-1_41.

Author's Publications

1. Kwong-Sak Leung, **Ka-Chun Wong**, Tak-Ming Chan, Man-Hon Wong, Kin-Hong Lee, Chi-Kong Lau, Stephen K. W. Tsui: Discovering Protein-DNA Binding Sequence Patterns Using Association Rule Mining. Nucleic Acids Research (http://nar.oxfordjournals.org/cgi/content/full/gkq500)

2. **Ka-Chun Wong**, Kwong-Sak Leung, Man-Hon Wong: Protein structure prediction on a lattice model via multimodal optimization techniques. In Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (Portland, Oregon, USA, July 07 - 11, 2010). GECCO '10. ACM, New York, NY, 155-162. DOI= http://doi.acm.org/10.1145/1830483.1830513

3. **Ka-Chun Wong**, Kwong-Sak Leung, Man-Hon Wong: Effect of Spatial Locality on an Evolutionary Algorithm for Multimodal Optimization. EvoApplications (1) 2010: 481-490

4. **Ka-Chun Wong**, Kwong-Sak Leung, Man-Hon Wong: An evolutionary algorithm with species-specific explosion for multimodal optimization. In Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (Montreal, Que'bec, Canada, July 08 - 12, 2009). GECCO '09. ACM, New York, NY, 923-930. DOI= http://doi.acm.org/10.1145/1569901.1570027