

“Machines take me by surprise with great frequency”

— Alan Turing

A Multi-Objective Genetic Algorithm with Side Effect Machines for Motif
Discovery

Farhad Alizadeh Noori

Computer Science

Submitted in partial fulfillment
of the requirements for the degree of

Masters of Science

Faculty of Computer Science, Brock University
St. Catharines, Ontario

© September 2012

To my parents, without whom I would not be here today.

Abstract

Understanding the machinery of gene regulation to control gene expression has been one of the main focuses of bioinformaticians for years. We use a multi-objective genetic algorithm to evolve a specialized version of side effect machines for degenerate motif discovery. We compare some suggested objectives for the motifs they find, test different multi-objective scoring schemes and probabilistic models for the background sequence models and report our results on a synthetic dataset and some biological benchmarking suites. We conclude with a comparison of our algorithm with some widely used motif discovery algorithms in the literature and suggest future directions for research in this area.

Acknowledgements

First and foremost I would like to express my deepest gratitude towards my supervisor Prof. Sheridan Houghten for her support and assistance throughout the course of my research and this thesis. Her kind words and insightful suggestions have always guided me in the right direction when I was lost in the sea of papers.

In addition, I would like to thank my thesis committee for their invaluable guidance and thought provoking questions. Especially, I would like to thank Prof. Brian Ross for his constant help and accommodation during my regular visits to his office. Also, Dan Ashlock for his immensely knowledgeable remarks and constant assistance. Furthermore, my friends in the computer science department: Martin, Mahmood, Mehdi, Arash, Iman, Tuhin, Francis and also throughout university: Yousef, Armin, Majid, Arghavan, Hedieh and others.

Last but not least I would like to thank my loving family, for their constant support through all I have ever done. Also, my girlfriend, Ghoncheh, for her patience and understanding during my studies.

F.A

Contents

1	Introduction	1
1.1	Biological Background	1
1.1.1	DNA and RNA	1
1.1.2	Motifs	4
1.2	Problem Statement	5
1.3	Challenges	5
1.4	Thesis Organization	6
2	Methodology	8
2.1	Evolutionary Computation	8
2.1.1	Introduction	8
2.1.2	Genetic Algorithms	10
2.2	Probabilistic Methods	18
2.2.1	Introduction	18
2.2.2	Markov Chains	19
2.2.3	Probabilistic Models for DNA Sequences	20
3	Literature Review	23
3.1	Motif Representation	23
3.2	Motif Discovery Algorithms	26
3.2.1	Consensus-Based Methods	26
3.2.2	Alignment-Based Methods	29
3.2.3	Evolutionary Techniques	33
4	Side Effect Machines	38
4.1	Definition	38
4.2	SEMs and DNA Classification	39
4.2.1	Dataset For Classification	40

4.2.2	Representation	41
4.2.3	Reproduction	41
4.2.4	Fitness Function	41
4.2.5	Experiments and Analysis	44
4.3	Restricted SEM for Pattern Recognition	47
5	Genetic Algorithm Design	49
5.1	Representation	49
5.2	Sexual Reproduction	49
5.2.1	Crossover	50
5.2.2	Mutation	51
5.3	Multi-Objective Fitness	53
5.3.1	Objectives	53
5.3.2	Scoring Method	55
5.4	Algorithm Outline and the Archive Set	56
6	Datasets	57
6.1	Synthetic Dataset	57
6.2	Tompa et al's Dataset	58
6.3	Metazoan Compendium	58
7	Results	60
7.1	Synthetic Dataset	60
7.1.1	A Note On Performance Measures	62
7.2	Tompa's Dataset	63
7.2.1	Objective Effectiveness Study	63
7.2.2	Comparison to other programs	64
8	Improved Results	67
8.1	Motif Database	67
8.2	Mass Experimentation Tool and Summary Page	67
8.2.1	Motif Representation	68
8.2.2	Ranking System	69
8.2.3	Background Model Additions	69
8.2.4	Multi-objective Scoring Schemes	69
8.2.5	Convergence Charts	69
8.2.6	Matches to JASPAR	70
8.3	New Experiments	71

8.3.1	IID vs Markov Models	71
8.3.2	Multi-objective Scoring Scheme Comparison	83
8.4	Comparison to Other Motif Finders	90
9	Conclusion and Future Works	97
9.1	Summary	97
9.2	Future Work	99
	Bibliography	106
	Appendices	106
A	Found Motifs	107
A.1	107
B	Convergence Charts	117
B.1	117

List of Tables

2.1	The Prisoner's Dilemma	10
4.1	Sample Count Vectors	40
4.2	Relative Entropy vs Entropy Experiment Parameters	44
4.3	Percentage of Bases	45
7.1	Parameters for Run on Synthetic Dataset	62
7.2	Objective Effectiveness Comparison	64
7.3	Found Motifs In Yeast04R	65
7.4	Found Motifs In Yeast08R	65
7.5	Found Motifs In Human03R	65
8.1	Background Model Experiment Parameters	72
8.2	Relative Entropy vs Entropy Experiment Parameters	86
8.3	Comparison to Other Algorithms Experiment Parameters	91
8.4	Results of Different Algorithms on the Metazoan Compendium	94
8.5	Comparison Results on the Metazoan Compendium	95
8.6	Comparison Results on the Metazoan Compendium	96

List of Figures

3.1	Representations of Motif	24
3.2	The “Creb” motif found in the human genome	26
4.1	SEM and Transition Matrix	39
4.2	Parents Before One-point Crossover	42
4.3	Offsprings After One-point Crossover	42
4.4	A Simple Two-State SEM	45
4.5	DNA Classification with Variable State Count	46
4.6	Detector of HNF-1 with 87% accuracy	47
4.7	Next or First State Side Effect Machine	47
4.8	DNA Classification with Variable State Count	48
5.1	Parent SEMs before the State At Index Crossover	50
5.2	Parent SEMs after the State At Index Crossover	51
5.3	Parent SEMs after Genetic Material Crossover	51
5.4	Parent Chosen for Mutation	52
5.5	Offspring After Random Transition Mutation	52
5.6	Offspring After Frontal Decay Mutation	53
5.7	Offspring After Posteriorl Decay Mutation	53
5.8	NFS-SEM with all transitions going to the next state	54
7.1	Average PWM divergence (30 runs)	61
7.2	Average Edit Distance (30 runs)	61
8.1	Snapshot of Program Output	68
8.2	IID background model convergence charts. The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood	74

8.3	Found motifs in the human CREB TF dataset using an IID background model. (a) and (b) are from the HSF1_Page dataset, (c) and (d) are from GATA_Pauli and (e) and (f) are from CREB_Zhang	75
8.4	1st-order Markov background model convergence charts. The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood	76
8.5	Found motifs in the human CREB TF dataset using an MM1 background model. (a) and (b) are from the CREB_Zhang dataset, (c) and (d) are from HSF1_Page and (e) and (f) are from GATA_Pauli	77
8.6	Comparison between JASPAR's reported motif for CREB_Zhang and found motif	78
8.7	Non-homogeneous 1st-order Markov background model convergence charts. The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood	79
8.8	2nd-order Markov background model convergence charts. The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood	81
8.9	Found motifs in the GATA_Pauli TF dataset using an MM2 background model. (a) and (b) are two found motifs and (c) is the JASPAR's reported motif for the dataset.	82
8.10	Found motifs in the CREB_Zhang TF dataset using an MM2 background model. (a) and (b) are two found motifs and (c) is the JASPAR's reported motif for the dataset.	82
8.11	Found motifs in the HSF1_Page TF dataset using an MM2 background model. (a) and (b) are two found motifs and (c) is the JASPAR's reported motif for the dataset.	83
8.12	3rd-order Markov background model convergence charts. The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood	84
8.13	Found motifs in the human CREB TF dataset using a MM3 background model. (a) and (b) are from the CREB_Zhang dataset, (c) and (d) are from HSF1_Page and (e) and (f) are from GATA_Pauli	85

8.14	SPEA2 convergence charts. The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood	87
8.15	NSGA-II convergence charts. The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood	88
A.1	Instances of motifs found in the Human_CREB dataset using the NSGA-II approach	108
A.2	Instances of motifs found in the Elegans_GATA dataset using the NSGA-II approach	109
A.3	Instances of motifs found in the Human_HSF1 dataset using the NSGA-II approach	110
A.4	Instances of motifs found in the Human_CREB dataset using the SPEA2 approach	111
A.5	Instances of motifs found in the Human_GATA dataset using the SPEA2 approach	112
A.6	Instances of motifs found in the Human_HSF1 dataset using the SPEA2 approach	113
A.7	Instances of motifs found in the Human_CREB dataset using the Summed Ranks approach	114
A.8	Instances of motifs found in the Elegans_GATA dataset using the Summed Ranks approach	115
A.9	Instances of motifs found in the Human_HSF1 dataset using the Summed Ranks approach	116
B.1	Convergence Charts for HNF1a_Odom. The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood	117
B.2	Convergence Charts for E2F_Ren. The objectives are as fol- lows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood	118
B.3	Convergence Charts for HSF1_Page. The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood	118
B.4	Myc_Oryan The objectives are as follows : Objective0 = En- tropy, Objective1 = Found Motif Count and Objective2 = Likelihood	119

B.5	ETS1_Hollenhorst	The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood	119
B.6	MEF2_Blais	The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood	120
B.7	SRF_Cooper	The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood	120

Chapter 1

Introduction

In this chapter some background information on biological terms and processes used throughout this work is discussed. Furthermore, the aims and goals of such a scientific effort and its importance and challenges are elaborated. This chapter is then concluded with an outline of the entire work.

1.1 Biological Background

1.1.1 DNA and RNA

Structure

Deoxyribonucleic acid(DNA) is a macromolecule which is made up of nucleobases which are nitrogenous bases bound to a sugar. Two different groups of nucleotides exist in DNA:

- Purines which include *Adenine*(A) and *Guanine*(G).
- Pyrimidines which include *Cytosine*(C) and *Thymine*(T)

The nucleobases bind together with hydrogen bonds. As depicted in Figure 1.1, Thymine binds with Adenine and Cytosine binds with Guanine. These base pairs are bound together in a strand with phosphate bonds which make up the DNA sequence (Figure 1.1).

Figure 1.2 shows two strands of nucleotides bound together in an *anti-parallel* double helix form. Anti-parallel means that the two strands run in opposite directions from each other with one strand complementing the other.

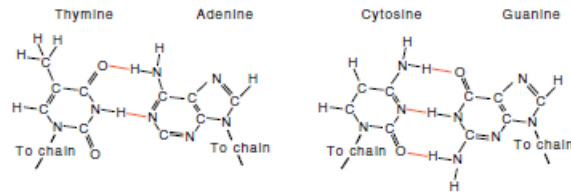


Figure 1.1: Base Pairs. Image extracted from [41].

This double strand makes up the DNA sequence which can be represented as a string of characters with possible alphabet $Alph = \{A, C, G, T\}$.

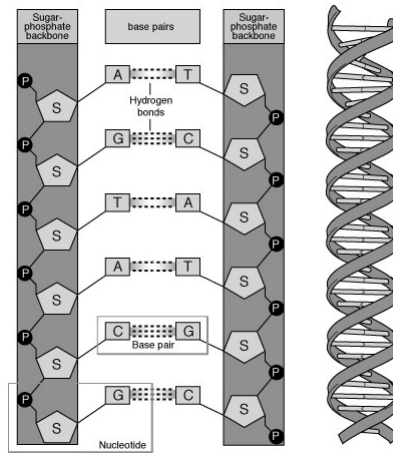


Figure 1.2: DNA. Courtesy: National Human Genome Research Institute.

A *Ribonucleic Acid*(RNA) is another molecule that when in form of *mRNA* is used to transfer genetic information. RNA has a very similar structure to the DNA with the difference that it has a different type of sugar and it has *Uracil*(U) instead of Thymine in the structure.

What the DNA Encodes

DNA encodes most of the information needed for the creation, functioning and reproduction of an organism. Extracting this information will give us the blueprint of any organism known to exist. Knowing the blueprint of an organism not only gives us insight into the concept of life but also helps us preserve life by understanding, repairing or reproducing the mechanisms of it.

More importantly, the secret to some human sicknesses or, to some extent, behavior is conserved inside the DNA and has been passed on from each generation to the next for years. One of the biggest problems is the decoding of this information. To extract this information we have to understand the “language” of the DNA.

Central Dogma of Molecular Biology One of the main focuses of biology is to understand how organisms inherit and mutate traits. These traits range from something unique like the color of a person’s eyes to something shared in all forms of a species like the production of antibodies for a disease. But how does this process start?

The sections of the DNA which code information are called *genes*. These genes code the information needed to start the chain reaction of events that would create the entity that the information encodes. The process of creation of this entity is called *gene expression*.

Proteins are the workhorses of life. Genes usually encode recipes for creation of proteins. Proteins created from expression of one gene could then participate in the expression of another gene and effectively the creation of other proteins. The process of creation of proteins from the DNA is known as the central dogma of molecular biology. This process has two main stages:

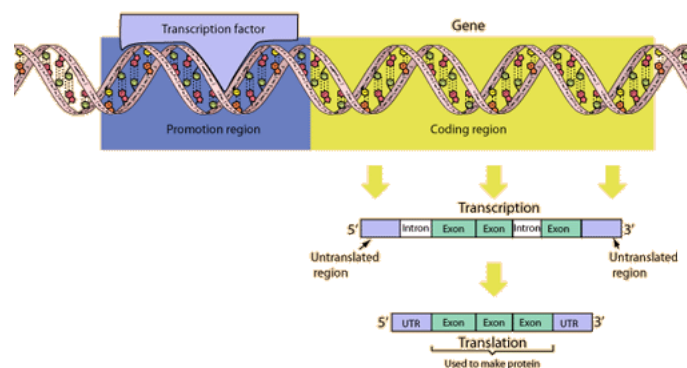


Figure 1.3: Transcription¹

Transcription

There are regions in the DNA that do not code for proteins. Initially

¹<http://hyperphysics.phy-astr.gsu.edu/hbase/organic/transcription.html>

these were dubbed “junk DNA” since they were believed to be useless. But some of these regions were later found to be quite important to the gene expression process by containing regulatory elements controlling the process. These regions, also called *regulatory regions*, control which genes are expressed and which are not. Regulatory regions are usually positioned upstream of the *gene transcription start site*. The start site is the position where the transcription begins by special proteins that bind to specific patterns in these sites. These regions with binding sites are also known as *promoters*.

The proteins initiating the transcription process are called *Transcription Factors* (TFs). Each transcription factor binds to a certain subsequence in the regulatory region with a certain affinity which also depends on the level of gene expression. The binding sites for TFs are quite diverse, i.e they could have variations or mismatches in certain positions.

In the transcription stage, the transcription factors form a part of the *Transcription Initiation Complex*(TIC) which starts the process. Also a part of this complex is an enzyme called *RNA polymerase*. When the TIC is activated, the RNA polymerase glides through the DNA sequence unwinding the double helix and creating a type of RNA called *messenger RNA* or *mRNA*. This mRNA would then find its way out of the nucleus into the cytoplasm.

Translation

The mRNA transcribed from the DNA is shown in Figure 1.3. This molecule contains two tails which signify either end of the mRNA. In translation, another complex called *ribosomes* binds to the mRNA and according to the information encoded in the mRNA, building blocks of proteins called *amino acids* are bound together in a string of amino acids which forms the protein. Each 3 consecutive nucleotides (called *codons* signify one amino acid. The ribosome glides all the way through the mRNA until it is signified by by a stop codon to stop. After termination a protein decoded from the DNA remains.

1.1.2 Motifs

Motifs are patterns of DNA sequence in short length (usually less than 30bps) that are of some biological significance. Binding sites for TFs are examples

of motifs. By recognizing each transcription factor's motif one may be able to understand and manipulate the expression of a certain gene.

Since motifs are patterns of short length, it is very hard to identify them by searching through the entire genome or even the promoter region which usually extends to thousands of base pairs long. Also, the promoter regions could have binding sites for different TFs, so a better solution is to look at promoter regions of a series of co-regulated genes. Since the genes are co-regulated (regulated by the same transcription factor) one may assume that the same *Transcription Factor Binding Site*(TFBS) of a given TF appears in the sequences more frequently. Another possible source of TF binding sites are sequences extracted from ChIP-chip experiments which are designed to biologically find the sequences that a TF binds to, an example of which is discussed in Section 6.3.

1.2 Problem Statement

The motif discovery problem could be formalized as the problem of finding strings of length m that are statistically over-represented in the input sequences $\{S_1, S_2, \dots, S_n\}$ where S_i is a subsequence from an organism's genome. The output is a set of candidate motifs (binding sites for transcription factors) that are thought to be present in the input sequences.

We focus on *de novo* motif discovery in this work, which requires an unsupervised approach to find *undiscovered* motifs in a set of input sequences.

1.3 Challenges

The motif discovery problem formulated above has some intrinsic complexities which include:

- **Lack of Locality:** Regulatory regions can be far away from the coding regions they regulate. This lack of locality in the search presents a problem to the search algorithm.
- **Relative Small Size:** Motifs have a very small size when compared to the size of the sequence containing them which could extend to millions of bases long.

- **Motif Degeneracy:** Motifs that encode the same conserving function are not always exactly the same and may have some variation to them.
- **Complexity:** The motif discovery problem has been shown to be a very computationally hard problem even when being the subset (alphabets restricted) of the equivalent computer science problem of approximate string matching which is NP-Complete [16].
- **Unidentified Motifs:** Even if the motif finder algorithm finds a pattern that is not an already known motif, it would take biological experimentation to show if it is really a novel unidentified motif or not.
- **No Best Single Objective:** The motif finding task does not have a single best objective to maximize. Although many algorithms only maximize the statistical likelihood of the pattern, the dataset, may also hold less likely patterns that are not known motifs.
- **No Widely Accepted Benchmarking Suite:** Although several benchmarking suites have been introduced in the literature, there is no one single accepted and widely used benchmarking suite on which to test the performance of algorithms. The datasets used for the task are usually either specifically curated by the authors or use benchmarking suites that are not widely used and consequently cannot be used to compare with other algorithms.

1.4 Thesis Organization

Our approach utilizes a multi-objective genetic algorithm to train *Side Effect Machines* (SEMs) , defined in chapter 4, which represent the motif. We first analyze the SEM functionality and performance by applying it to a DNA classification problem. The SEM is later restricted to make the motif extraction process from the SEM straight-forward and a restricted version of the SEM, called a *Next or First State Side Effect Machine*(NFS-SEM), is defined. We first test the performance of different suggested objectives to the problem on a synthetic and a biological dataset. We then test the program on a more recent benchmarking suite containing a set of biological datasets, finding comparable results to the best motif finding algorithms in the literature.

The remainder of this thesis is organized as follows: Chapter 2 contains definitions of some of the methods used in the motif discovery field. Chapter 3 contains a review of the literature on motif discovery algorithms. In Chapter 4, SEMs and NFS-SEMs are defined and initial experimentation on the DNA classification problem are reported. Details on our motif discovery algorithm design is discussed in Chapter 5. We provide information on the datasets used to test our algorithm in Chapter 6. Chapters 7 and 8 contain results on these datasets. Chapter 9 presents conclusions and future directions.

Chapter 2

Methodology

In this chapter an overview of the methods used for the motif discovery problem is given. In the first section a brief description of the *evolutionary algorithm* used is given and in the next section some probabilistic methods used in the algorithm are discussed.

2.1 Evolutionary Computation

2.1.1 Introduction

In science most innovation comes from adding newer ideas to already existing concepts. Any scientific contribution is built upon work done by others before. On the other hand just as adding and modifying existing ideas has been prevalent in the past, a newer approach tries to find solutions to problems by studying how nature approaches its problems. This area is called *bio-inspired computation* [18]. The swarming of ants and bees, for instance, are examples of this type of algorithm. Ants and bees when working together are able to accomplish complicated tasks. These tasks are done by delegating simpler tasks to individuals in the population, thus breaking the problem down into smaller ones solvable by a single individual. Also these individuals communicate with one another about, for example, a source of food or a place to avoid, which provides a perfect analogy for the problem of search in computation.

In order to mimic the problem-solving mechanisms in nature, one has to know how nature has learned to carry out these tasks. The answer to this

of course is evolution. Evolution is the adaptation of a species to the environment so as to maximize its chances of survival. Many times in the face of pressures from the environment, individuals in the species that are not able to adapt die out and the “fitter” individuals survive and reproduce. This process throughout generations creates offspring that are ever fitter with respect to likely pressures from the environment. But how does nature generate the diversity needed to overcome environmental challenges?

Crossover and mutation are two methods used in reproduction to create variation and to test newer individuals that may be fitter. Crossover attempts to mix the genetic material of the parents by inheriting certain genes from one parent and others from another. Likewise, mutation makes changes to the organism’s DNA and effectively changes the way the organism functions. But the question is still unanswered. Why are these changes for the best and the children fitter?

From a computational point of view, the answer is that nature is just doing a search. It is important to understand that nature is not necessarily searching for the “best” solution to the problem but only a solution that works. Hence the children that are produced are experiments carried out by nature in the *fitness landscape*(space of all possible solutions) of life. Fitter individuals usually have higher chances of survival and reproduction. After the offspring is born if the reproduction methods created a fitter individual, the offspring survives and itself reproduces. Conversely, if the individual is not able to cope with environment pressures, other fitter members of this newer generation would survive and reproduce. This idea is the basis of *evolutionary computation*(EC). The individuals in EC are represented as a two-layer structure which consists of:

- Genotype: This represent the genetic code, i.e the genes that encode information about the individual and are the inputs to the reproduction operators.
- Phenotype: This represents the expression of the genes or the behavior of the individual when interacting with the environment.

Evolution in EC starts with a population of individuals. Each individual has a genotype and a phenotype. The individuals in the first generation are created either in a supervised or random manner. This first generation goes through the evaluation phase in which each individual’s fitness is calculated. The fitness is extracted with regards to the individual’s phenotype. Next

comes the reproduction phase in which the fittest individuals have priority. The biological equivalent of this rule is “natural selection” or “survival of the fittest”. In the reproduction phase either crossover or mutation creates offspring from the fitter individuals. The aim is to improve the average fitness of the generations through the iterations, mimicking *Darwinian evolution* (evolution based on heritable differential reproductive potential). Different termination criteria exist for EC, including a prespecified number of generations or a specified number of generations with no improvements.

2.1.2 Genetic Algorithms

Genetic algorithms (GA)[20] are a category of EC in which individuals are commonly represented by bit strings. Just like EC individuals in a generation compete and evolve using the same principles. The bit string represents the genotype and the interpretation of the bit string is the phenotype from which the fitness is calculated. An example of an interpretation could be the binary representation of the gene.

Case Study

We will use a case study to show how a problem can be solved with a GA. In each section we present how the concept should be implemented for the problem. We will try to evolve a GA that is able to play the game of *Prisoner’s Dilemma*.

In this game two opponents are presented with two options: “cooperation” and “defection”. If opponents cooperate they are both paid off; if one defects then the player who cooperated gets nothing and the one who defected gets a higher pay off; if both players defect they receive a minimal pay off. This is summarized in Table 2.1.

Table 2.1: The Prisoner’s Dilemma

Player	B(Cooperate)	B(Defect)
A(Cooperate)	3/3	5/0
A(Defect)	0/5	0/0

Human players usually learn to cooperate as the game is played in more rounds to get more points each. On the other hand, a solution to the prob-

lem using game theory is minimizing the maximum damage the other player can inflict. This means that the right action would be to defect. Another strategy is to use the “tit for tat” method which would cooperate in the first round and in the next rounds copy the exact action the other player made, i.e if the other player had defected the player would defect and similarly for the other cases. Axelrod and Forrest as reported in [22] developed a GA that evolved the “tit for tat” method which is explained throughout this section. They and also developed an improvement to realize if the opponent could be fooled into cooperating in the face of defection; if not, it would resolve to “tit for tat”. This improvement is not discussed here.

The representation of the GA should encode the information about the strategy the algorithm takes for the game. We assume that the decision for the outcome of the algorithm is based on the last 3 rounds of the game. Since each player has 4 options in each round and we have 3 rounds in history we need to be able to store 64 possible outcomes. For example the first gene would represent the outcome of the algorithm for three mutual defections. The other 63 bits are setup for all possible permutations of outcomes and the number 0 or 1 would be the decision made. The all zero genome then would be defection in all cases.

Initialization

In the initialization step the individual is first initiated by specifying a length for the chromosome. Then, the operation continues by randomly selecting 0 or 1 using a coin flip until the individual is filled with 0s and 1s. This phase may also use auxiliary problem-dependent information which would help the evolution by supplying already known fit individuals.

In the prisoner’s dilemma case the initial individuals (bit strings of size 64 bits) are randomly created by putting 1 or 0 in each bit of the bit string.

Evaluation

Fitness evaluation is a phase in the genetic algorithm’s evolution where all individuals’ fitness in the current generation are evaluated. Usually the individuals have a single property or objective that differentiates them in terms of fitness score.

The fitness value in our case study is the average number of payoffs an individual’s encoded strategy received after some number of played games.

Selection Methods

Tournament Selection This method requires only one parameter which is the size of the set of individuals s chosen to be considered for selection. Initially s individuals are randomly chosen from the population. Then depending on the reproduction method the one or two fittest individuals are chosen and returned. This method ensures that fitter individuals are selected on average. The parameter s is also important to consider. A higher value for s puts higher selection pressure on the generation and lower values of s put less selection pressure. In the extreme case where $s = 1$, the selection is just random selection resulting in random search.

Roulette Wheel In this method the fitness value for the individuals is normalized and the individuals are selected proportionate to their fitness values as in a roulette wheel where each individual takes up proportionate space on the wheel. This method also ensures that more fit individuals get selected on average.

Roulette wheel selection is more commonly called fitness proportional selection in more recent work.

Genetic Operators

Crossover

One-Point Crossover As an example we take the case of the bit string for the individuals' genotype in our case study. In this type of crossover a position in the gene is first selected uniformly at random from the two parents. This would divide the two parents into two segments. The segments in these parents are then swapped. The swapping could be done either by exchanging the segment to the right of the pivot point in both parents or any other combination.

Two-Point Crossover Assuming a bit string representation for the genotype, in this type of crossover two random positions in the bit string are selected in the two parents. The part of the genotype between these two points are then swapped between the parents.

Mutation The mutation operator could be designed to change the genotype dramatically or be thought of as corrections in random places in the genotype. In the former case, for example, one could go through all the bits of the bit string and choose to flip the bit at that position by a coin flip (0.5 probability). In the latter case a random position in the gene is selected for mutation and the bit is flipped.

Replication This operator copies the chosen individual from the selection phase and copies it to the next generation without any modification.

Elitism

A common problem with GAs is that very good solutions evolved at the early stages of the generational run may be eliminated due to destructive reproduction operators. Therefore, to overcome this issue a certain number of best individuals in each run are copied without modification to the next generation to preserve them from destruction.

Multi-objective Fitness Evaluation Strategies

The prisoner's dilemma implementation discussed previously aims to improve one objective which is the average payoffs over a number of played games. However, there exist other problems that aim to improve more than one objective. As an example identification of cats in an image is a multi-objective problem since using a single objective of "*Number of Cats*" or "*True Positive*" (TP) could easily and prematurely evolve individuals which recognize every single object as cats in the image. The reason for this is that this approach would give better scores to individuals that recognize every single object as a cat than individuals that only get the real cats as answers. Consequently, another objective is needed to differentiate these two groups. This objective could be the "*Number of Non-cats*" or "*True Negatives*" (TN) which would balance the identification of the two groups. Some strategies that received more attention at the time of early efforts in the area of multi-objective fitness evaluation include [19], [23], [45] which all used a non-dominated sorting algorithm coupled with a diversity strategy. It was shown afterwards in [51] that elitism has a positive effect in multi-objective evolutionary algorithm's evolution. In this subsection firstly classic strategies are discussed and then the two most popular methods are elaborated.

Weighted Sum : Many different strategies can be thought of to improve more than one objective simultaneously, which could range from simple ones like adding or averaging objectives to constructing complicated problem specific formulas with weights for each. These methods if chosen fail to account for the improvement of one objective at the expense of the other since they do not enforce any restrictions. And this is more than often the case where one objective works against the other one or two objectives and they both cannot improve concurrently. Furthermore, all these methods lack a strategy to encourage diversity or convergence to the “*Pareto Optimal Set*”.

Pareto Optimal Set : This is the set of all individuals in a multi-objective problem that are not dominated by any other individual. An individual is said to dominate another if and only if it is better in all objectives. This definition leaves the domination criteria undefined for two individuals where one is better than the other in some objectives and worst in some others. These two individuals cannot be compared and are considered to be in the same set.

The Pareto Optimal Set is the set of all the individuals in the population that are undominated by any other individual. In a multi-objective optimization problem with n objectives, Individual $A(a(1), a(2), \dots, a(n))$ is said to dominate individual $B(b(1), b(2), \dots, b(n))$ if and only if:

$$\forall i : a(i) \leq b(i) \wedge \exists i : a(i) < b(i)$$

This idea has been used in the field of GAs and multi-objective optimization previously [20]. These individuals are also called the “*Pareto Front*” or rank 1. These could later be removed from the population in an iterative process to find higher ranks in a process called “*Pareto Ranking*”. The problem with this method of ranking is that the Pareto Front is susceptible to outliers which would be of considerable distance to the other individuals in the archive. Since one cannot label any of the individuals in the archive as the best solution, one would want these solutions to produce a spectrum of all possible solutions spread out in the solution space with equal distances between them.

NSGA-II : This algorithm was introduced after some success of the previous algorithm by the the same authors in [45]. This approach is called the *Non Dominated Sorting Genetic Algorithm*(NSGA). The creators report that the previous algorithm had the following shortcomings:

- Expensive sorting algorithm : The sorting algorithm used was of order $O(MN^3)$ where M is the number of objectives and N is the number of individuals. This made the sorting algorithm too slow for big populations.
- Lack of Elitism : The authors reviewed the results of recent studies and deduced that elitism could help improve the performance of the algorithm by preserving the best individuals and preventing them from being lost.
- Extra parameters : the previous algorithm required an extra parameter which required fine tuning by the user.

In the improved version the algorithm uses another sorting method which takes $O(MN^2)$ time to complete. The sorting is done as follows: for each individual in the population calculate these two values:

- n_p : Number of individuals that dominate the current individual.
- s_p : A set of individuals that the current individual dominates.

The individuals with s_p equal to zero are then in the non-dominated front. Now for each of these individuals each member of their s_p is visited and their n_p is reduced by one. Lastly the recognized undominated set is removed from consideration and the algorithm iterates.

The diversity strategy of NSGA-II uses an estimate called the *Density Estimate*. This gives an estimate of how dense the solutions are at a certain point the Pareto front. The estimate is calculated by first sorting the solutions in the front by each objective in ascending order and then calculating the summation of the normalized difference between the objectives from the individual before and after each solution. This gives an estimate of how far the two closest individuals are from the current one. The fitness comparison between individuals then is as follows: if the two individuals are in different Pareto ranks then the individual with the lower rank is preferred but if the individuals are in the same rank, the one in the less crowded area is preferred which means less density.

SPEA2 : The *Strength Pareto Evolutionary Algorithm*(SPEA) algorithm also found a successor after the introduction of NSGA-II and its dominance over SPEA. SPEA2 was introduced a year after NSGA-II in 2001. The authors report the differences as follows:

- Improved fitness assignment which takes into account the number of individuals dominating or dominated by a certain individual.
- A density estimation technique based on a nearest neighbor strategy.
- A new archive truncation method to preserve boundary solutions.

The SPEA algorithm starts with an empty archive and a regular population. At the time of evaluation first the Pareto front is chosen from the population and copied to the archive. If there are any individuals already in the archive that are dominated by the new individual they are removed, also duplicates which are individuals with the same objective values are replaced by the entering individual. If the size of the archive after the addition is bigger than the specified value then the archive is reduced to its right size using a clustering technique described below.

Firstly the fitness of each individual in the archive is calculated by calculating a strength value $S(i)$ (will be regarded as fitness value) which is the number of individuals in the population that are dominated by it or are equal to it (in the sense of objectives) divided by the population size plus 1. Then the fitness value of the individuals in the population that are not in the archive is calculated. Their fitness would be the summation of the strength values of all the members of the front that dominate or are equal to the current individual. This means that each individual is penalized if it is dominated by more individuals in the front. Also notice that fitness is to be minimized in this case since the archive individuals have lower fitness values. The mating process would then choose individuals from the union of the archive set and the population with a binary tournament selection. Finally the old population is replaced by the new offspring.

Potential weaknesses of this approach are for example the situation in which the archive holds only one individual which effectively makes all other individuals have the same fitness value since all other individuals are either equal or are dominated by that individual, which in turn changes the algorithm to random search. The fitness evaluation in SPEA2 is the same as SPEA with the difference that instead of only taking into account the individuals in the archive to calculate each individual's fitness, other individuals in the population that dominate it are also included. Also the reproduction phase differs in the sense that the individuals in the archive are only used for reproduction whereas in SPEA both the archive and population took part in reproduction.

The additional density information added with SPEA2 uses the k nearest neighbor method to estimate how close the the k th neighbor is to the current individual. The higher the distance the better the fitness for the individual. Another improvement made is the archive truncation scheme. SPEA's archive does not have a size limit. In SPEA2 a size limit is added which means that at any point in the algorithm there are two situations possible:

- Archive's content not up to capacity: To fill up the archive other individuals sorted by their fitness from the next Pareto rank are added to the archive. This procedure iteratively continues until the archive reaches capacity.
- Archive's content exceeds capacity: In this case all the archive's individuals are sorted according to their distance to the closest neighbor and then individuals with the smallest distance to the next individual are eliminated. Ties are broken by calculating the distance of each individual to the second most closest etc.

SPEA2 was tested on certain test problems comparing it to some other techniques and it was shown to have a better performance than SPEA on them.

Summed Ranks : Summed rank introduced in [6] is a simple multi-objective evaluation technique which was initially shown to work well in problems with a high number of objectives. It was later shown that it could be effective in problems with few objectives too [38], [8].

In this method for all objectives, individuals are sorted in decreasing order and ranked. Individuals with equal or similar (problem specific) are given the same rank. This would yield a vector of ranks for all objectives of each individual. This vector represents the individual's fitness relative to others in each specific objective. This vector is then summed up to represent the individual's raw fitness. The lower the summed rank, the fitter the individual is deemed to be.

Since certain individuals can have the same value for an objective they should have the same rank. This could result in a bias towards objectives with a higher variability in their values, since by taking up more values they are able to climb up to a higher number in ranks and contribute more with the variation to the fitness. In order to remedy this problem objectives could be normalized. The normalization is done by taking the maximum and mini-

mum value (always 1) in each objective and calculating the following:

$$fitness = \sum_{allObjectives} \frac{ObjectiveValue}{MaxValue - MinValue}$$

2.2 Probabilistic Methods

2.2.1 Introduction

Most of the studies done in statistics have been about independent trial processes. These processes form the basis of classical probability theory.

When a sequence of trial experiments are conducted with one following the other and they follow an independent trial process, the possibility outcome of any of the trials is the same and it has the same probability. In other words, the outcome of a certain trial in the chain of trials does not influence the probability of the experiments after it. An example of an independent trial process is the act of continuously flipping a coin. The outcome of each of the trials is always $S = head, tail$ and also the probability of any of the possible outcomes is always 0.5. Each of these experiments can be thought of as a random variable and the set of all these random variables are said to be *Independent and Identically Distributed*(iid).

The probability of a sequence of trials could then be calculated by the chain rule of probability as follows:

If we denote each experiment with S_i where $1 < i < n$, then according to the conditional independence of experiments and according to basic rules of probability we have:

$$\begin{aligned} P(S_1, S_2, \dots, S_n) &= P(S_n|S_{n-1}, \dots, S_1) \cdot P(S_{n-1}|S_{n-2}, \dots, S_1) \dots P(S_2|S_1) \cdot P(S_1) \\ &= P(S_1) \cdot P(S_2) \dots P(S_{n-1}) \cdot P(S_n) \end{aligned} \quad (2.1)$$

In modern probability theory these processes were extended to include knowledge from past experiments. For example when trying to predict a student's grade in the final exam one might use information about his midterm and assignments' scores. In 1907 A. A. Markov started the study of these types of processes. Hence these processes are called Markov chains. A good review of the use of different probabilistic models for biological sequence analysis can be found in [1].

2.2.2 Markov Chains

A Markov chain consists of a series of states, $S = s_1, s_2, \dots, s_r$. The chain starts at one of these states and makes transitions from one state to another. The starting state is determined by a probability distribution defined on S . Each of the state transitions are called a *step*. The probability of a certain transition from state s_i to state s_j is denoted by p_{ij} . If this probability depends on the time s_i the Markov chain is called a *non-homogeneous Markov chain* or a *non-stationary Markov chain*. If p_{ij} stays the same regardless of the time or the state the transition was made from then the Markov chain is called *stationary* or *time-homogeneous*.

First Order Markov Chain

The Markov chain should have a property called the Markov property. This property states that at any certain state the probability of transition to future states is only dependent on the current state and not on the states before. In a *first-order Markov model* this property holds.

The probability of an occurrence of a sequence of experiments following a first order Markov model assuming a graphical model of which the Markov property enforces, can be calculated as follows:

$$\begin{aligned} P(S_1, S_2, \dots, S_n) &= P(S_n|S_{n-1}, \dots, S_1) \cdot P(S_{n-1}|S_{n-2}, \dots, S_1) \dots P(S_2|S_1) \cdot P(S_1) \\ &= P(S_1) \cdot P(S_2|S_1) \dots P(S_n|S_{n-1}) \end{aligned} \quad (2.2)$$

Example 2.2.1. Assume that we have a transition matrix P and an initial probability vector I as below for weather changes and also assume that the transitions follow the Markov property. We want to calculate the probability of having the weather sunny today, rainy tomorrow and snowy the next day.

$$P = \begin{array}{c} \text{Transition Matrix} \\ \begin{array}{ccc} \text{Sunny} & \text{Rainy} & \text{Snowy} \\ \text{Sunny} & \left(\begin{array}{ccc} 0.25 & 0.30 & 0.45 \\ 0.15 & 0.50 & 0.35 \\ 0.60 & 0.30 & 0.10 \end{array} \right) \end{array} \end{array}$$

$$I = \begin{array}{c} \begin{array}{ccc} \text{Sunny} & \text{Rainy} & \text{Snowy} \\ \left(\begin{array}{ccc} 0.65 & 0.20 & 0.15 \end{array} \right) \end{array} \end{array}$$

Solution The probability of (*Sunny, Rainy, Snowy*) is as follows:

$$P(\text{Sunny}) \cdot P(\text{Rainy}|\text{Sunny}) \cdot P(\text{Snowy}|\text{Rainy}) = 0.65 \times 0.30 \times 0.35 = 0.06$$

This means that there is a 6% chance that this sequence happens one after the other.

Second and Higher Order Markov Chains

In these types of Markov chains each type of transition for future states depends on the current state and the state before.

As an example, the probability of occurrence of a sequence of experiments following a second order Markov model can be calculated as follows:

$$\begin{aligned} P(S_1, S_2, \dots, S_n) &= P(S_1|S_2, \dots, S_n) \cdot P(S_2|S_3, \dots, S_n) \dots P(S_{n-1}|S_n) \cdot P(S_n) \\ &= P(S_1) \cdot P(S_2|S_1) \cdot P(S_3|S_2, S_1) \dots P(S_n|S_{n-1}, S_{n-2}) \end{aligned} \quad (2.3)$$

2.2.3 Probabilistic Models for DNA Sequences

DNA sequences can be modeled as a trial of experiments each having a probability of occurrence. The probability of the sequence depends on the probabilistic model assumed for the DNA sequence.

Let us assume sequence $S = \text{“ACGGTGCGTAGTCAT”}$ is the DNA sequence that we want to model. If we assume that this sequence was created using 15 experiments which were all independent of one another and used the same probability distribution, we have actually modeled it using a iid probabilistic model. In other words, each position in the sequence can be thought of as a random variable which has 4 possible outcomes (A, C, G, T) and the probability of any of the possible outcomes remains constant between experiments.

There are two scenarios possible when one works with probabilistic models:

1. Parameter Fitting

In this scenario, we do not know the parameters of our probabilistic model (in the above example the probability vector for possible outcomes of the random variables). We would then try to fit parameters of the model by estimates like the *Maximum Likelihood Estimate* (MAP) which tries to maximize the likelihood of the observed data given the model parameters. In the above

examples calculating the frequencies of each of the nucleotide bases gives us a MAP estimate of the parameters of an iid model for the sequence. The MAP estimates if each base would be calculated by the formula:

$$frequency = \frac{n_i}{N} \quad (2.4)$$

where n_i is the number of occurrences of base i where $1 < i \leq 4$ and N is the sequence length.

If the assumed model is a first order Markov model then the MAP estimates could be calculated by first calculating the iid model since it will be used in the formula and then by calculating:

$$frequency = \frac{n_{ij}}{\sum_{j=1}^4 n_{ij}} \quad (2.5)$$

Where n_{ij} is the number of occurrences of base j right after base i where $1 < i \leq 4$ and $1 < j \leq 4$.

For a second order Markov model both the above frequencies have to be calculated plus the formula:

$$frequency = \frac{n_{ijk}}{\sum_{k=1}^4 n_{ijk}} \quad (2.6)$$

Where n_{ijk} is the number of occurrences of base k right after base i and j where $1 < i \leq 4$ and $1 < j \leq 4$ and $1 < k \leq 4$.

2. Calculating Likelihood of Observations

If the parameters of the probabilistic model are already available or calculated then the likelihood of a given observation can be calculated using either of the formulas calculated above.

There is a problem inherent in this calculation. What would one do if the observation holds a pattern that was not seen in the modeling stage? A naive answer to this question is assuming a 0 probability of such pattern. But that answer is not correct, the reason of which can be explained using the below example introduced by Laplace.

Example 2.2.2. What is the probability of the sun not rising tomorrow?

Solution This question although seeming philosophical reveals an intricate complication in calculating the likelihood of unseen events. In order to simplify the calculations we only consider the data from the past week for our modeling process.

We represent the rise of the sun by the binary random variable $R = t, f$ which can take the value t representing that the sun rose on that day or f the opposite. Now according to the example description we have seen the “ $tttttt$ ” pattern occur and we would like to find the probability of “ $ttttttf$ ”. As another assumption to simply the model we consider the rise of the sun to be statistically independent of the previous day enabling us to consider the events of the past week as 7 independent Bernoulli trials. Given this assumption we can calculate the probability of the patterns as:

$$P(X) = (1 - x)^{N-m}x^m$$

where X is the pattern and x is the probability of t or f .

Since we do not have the probability of sun rising tomorrow we would have to estimate it from the current data. We use the maximum likelihood estimator and find t and f as $P(t) = \frac{1}{7}, P(f) = \frac{0}{7}$ which is an overfit to the model and would result in a probability of 0 for the pattern “ $ttttttf$ ”. In order to fix the estimator *Laplace smoothing* or *1 additive smoother* is used. In these types of smoothing 1 is added to the numerator and denominator for each participating term. In this case since the formula has one term in the numerator (Number of times sun did not rise) and two formulas in the denominator (number of times sun didn’t rise + number of times sun rose) we add 1 to the numerator and 2 to the denominator:

$$P(t) = \frac{7 + 1}{7 + 2} = \frac{8}{9}, P(f) = \frac{0 + 1}{7 + 2} = \frac{1}{9}$$

So the probability of the sun not rising in the 8th day when 7 days have been observed is $\frac{1}{9}$. Notice that this number approaches zero as the number of observed days increases.

Chapter 3

Literature Review

In this chapter a review of previous work in the field of motif discovery is presented. This review demystifies some of the jargon used in the field and also analyzes in detail some of the more extensively used motif discovery tools in the literature. An overview of techniques using an evolutionary approach is also provided.

According to Sandre and Drablos [40] one of the early origins of DNA motif discovery is a computer program written in 1977 by Korn et al [29]. This program was able to discover similarities in sequences upstream of the binding sites. This program allowed mismatches and gaps but only allowed pairwise comparisons. Later on Queen et al [36] added to the work by enabling multiple sequence comparisons between sequences. Stormo et al [48] introduced a Perceptron algorithm using a sum of weighted position matchings of the motif and the sequence. Staden [46] introduced the position weight matrix which is mostly used in probabilistic methods and uses weights corresponding to log-frequencies of the bases in aligned matches of the motif and the sequence. In the following sections some of the more popular motif finders and motif representations are discussed.

3.1 Motif Representation

Several different representations of a motif have been suggested in the literature. A motif could be shown as a *consensus*. For example, the HNF-1 TF's binding site can be shown as the pattern "TATTGTTTATT". This shows how the motif is mostly in most occurrences but fails to show its degenerate

ACCTGAG		1	2	3	4	5	6	7
AGCGGAG	A	5	0	0	0	0	5	2
AGCTGGT	C	1	4	5	0	0	0	0
ACGTTAA	G	0	2	1	2	4	1	3
CCCTGAA	T	0	0	0	4	2	0	1
ACCGTAG								

(a) All occurrences
(b) Alignment Matrix

	1	2	3	4	5	6	7
A	0.83	0	0	0	0	0.83	0.33
C	0.17	0.66	0.83	0	0	0	0
G	0	0.34	0.17	0.17	0.66	0.17	0.50
T	0	0	0	0.66	0.17	0	0.17

(c) Frequency Matrix

Figure 3.1: Representations of Motif

forms. To show these other forms IUPAC symbols may be used. IUPAC symbols add more possible alphabets to the existing 4 bases. These alphabets add more flexibility to representation of the consensus sequence. For example the letter Y represents C or T and W represents A or T (U). Although more flexible, these forms still fail to show the frequency of occurrence of such bases.

Also, *Alignment Matrices* or *Position Weight Matrices* (PWMs) are another representation of a motif. For a motif of size m , the corresponding PWM is a matrix A_{ij} of size $4 * k$ whose columns $A_j, 1 \leq j < m$ represent location j of the motif. Each A_{ij} shows the number of occurrences of each nucleotide in each position of the motif instances.

The PWM may also be normalized to show the occurrence probability of nucleotides at these positions. These possible representations are depicted in Figure 3.1. In part (a) of this figure all found instances of a motif are listed which could then be clustered into a consensus sequence or alignment matrix (or PWM). The alignment matrix corresponding to the occurrences in part (a) is shown in part (b). Part (c) shows the normalized PWM or the frequency matrix which can be calculated from the alignment matrix by normalization.

Another method of motif representation uses the concept of entropy of a

certain base assuming each base in the motif to be produced by a statistical sampling method. Each location is considered to be a random variable that can hold 4 values (A,C,G and T). These random variables are considered independent and identically distributed assuming a multinomial distribution. The sampling process at each position samples from this multinomial distribution. The entropy score of the random variable is the amount of uncertainty about the outcome of that random variable before the sampling is done. The formula used for entropy uses the same concept as Shannon's entropy used in information theory [47]. The entropy score for location j is then calculated as follows:

$$E(X_j) = \sum_{i=1}^4 \frac{x_{ij}}{n} \log_2 \frac{x_{ij}}{n} \quad (3.1)$$

where n is the number of occurrences of the aligned motif and x_{ij} is the number of times letter i occurs in position j of the motif. In the case of DNA sequences i can take on 4 values representing $Alph = \{A, C, G, T\}$.

The value of the entropy of the variable is maximized when the uncertainty about the outcome of the variable is maximized, i.e. when all bases have an equal probability of occurrence. This would result in a value of 2 for the entropy. Also the value is minimized when the uncertainty is lowest and one base has the probability of 1 and others are 0s. In other words in all occurrences of the motif, only one base appears in position j . The entropy score in this case would be of value 0 which means no uncertainty. These entropy scores can then be used to create *sequence logos*, stacks of letters in each sequence position with letters proportioned according to their probability of occurrence. In order to translate the entropy score shown in Equation 3.1 to a proportional representation where the most common base has the highest proportional value the formula has to be changed as follows:

$$E(X_j) = 2 - \sum_{i=1}^4 \frac{x_{ij}}{n} \log_2 \frac{x_{ij}}{n} \quad (3.2)$$

The last form of motif representation discussed is the *motif logo*[42]. This form of representation of the motif stacks all possible letters at a position on top of one another. The size of the stacked letters are relative the their frequency or to the entropy as calculated in Equation 3.2. An example of a sequence logo representation is shown in 3.2. In this figure at position 0,

T is shown to be the most frequent letter to appear. Also, position 7 does not have a dominant letter with high frequency: while C has the highest frequency, A also takes its fair share of occurrence in the instances. The more frequent a letter, the bigger it is in the logo. Motif logos are a very intuitive form of visualization and are usually chosen to represent motifs in motif finder algorithms.

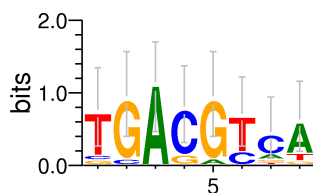


Figure 3.2: The “Creb” motif found in the human genome

3.2 Motif Discovery Algorithms

More than a hundred algorithms have been devised to tackle the single motif discovery problem. These algorithms can be divided into two broad categories: (i) enumerative or word based algorithms and (ii) alignment or profile based algorithms.

Enumerative algorithms search the entire space of possible n -mers for a motif of length n containing a preset number of mismatches. An exact match of n -mers to motifs is not desirable and some degeneracy should be accounted for by the algorithm.

3.2.1 Consensus-Based Methods

Some examples of word based methods are described below:

YMF

YMF [43] represents the motifs as strings over the alphabet A,C,G,T,Y,S,W,N which are IUPAC codes. IUPAC alphabets were discussed in Section 3.1. YMF (Yeast Motif Finder) was designed to find motifs in the promoter regions of *Saccharomyces cerevisiae*. YMF’s inputs are (i) the input promoter

sequences (ii) the number of spacers in the motif (places with N's) (iii) the number of non-spacers (any IUPAC model but N) in the motif and (iv) a position weight matrix containing base compositions of already computed background frequencies in the promoter regions of the yeast genome when assuming a third order Markov model for the background model. YMF finds motifs by ranking all possible words of length k (size of motif) by the z -score calculated by the formula:

$$z_s = \frac{N_s - E(X_s)}{\sigma(X_s)}$$

where N_s is the number of occurrences of the candidate motif in the sequences and $E(X)$ is the expected number of times given the background model. This formula calculates how many standard deviations the actual number of occurrences of the motif is from the expected value. The higher the value of the z -score the more likely it is that the motif was not produced from the calculated distribution of the background model. Since YMF assumes the current input sequences to be a sample from a background distribution of a third order Markov model with the given stationary distribution (the input position weight matrix) it has to check the p -value of the calculated statistic in order to make sure the high z -score did not happen by chance. The p -value measures the probability of a motif getting a score more extreme than the current score. To this end, 100 more datasets of the same dimensions are created from the background model and YMF is run on them. An estimate of the p -value is then calculated by deriving the fraction of times the z -score was greater. The motifs that have the highest z -score and are statistically significant are then reported.

Weeder

Weeder [34] is another enumerative technique which uses a consensus based representation of the motif. Weeder first constructs a suffix tree from the set of input sequences. The suffix tree is created by iteratively adding sequences one by one to the tree and also marking each suffix as coming from a certain sequence. This marking is done by a n -bit vector which is 0 in all places except the sequence indices it came from. To find a certain subsequence one can start from the root and continue along the path specified by the subsequence and eventually arrive at a marker vector which shows

the sequences that contain the subsequence. A subsequence matching which contains mismatches could use the same mechanism, with the difference that different paths could be taken in the sense that if the current character in the path doesn't match the subsequence character one mismatch is added to the number of mismatches and pattern matching is continued. If the number of mismatches reaches a predefined number the pattern is abandoned. Weeder uses a similar technique.

The algorithm starts from the root and checks if the pattern A is in more than q sequences. If so the pattern is expanded and the pattern AA is checked. If this pattern was not found in q sequences then one is added to the number of mismatches and AAA is checked. If no more mismatches are allowed AAC , AAT , AAG are checked similarly and are expanded. Basically all possible paths are checked in the suffix tree until the search is exhausted. Notice that the pattern that is being checked does not have to necessarily be a subsequence from the input sequences. A problem with this approach is that right from the start of the matching process all sequences of length e (possible number of mismatches) are possible paths to expand. But all these possible paths (3^e) already contain 4 mismatches and are very unlikely to produce a valid motif candidate. These patterns would be weeded out. As a means of eliminating the need for supplement of the mismatch count parameter from the user, Weeder uses an error ratio instead of a fixed number of possible mismatches. The error ratio is a function of the motif length. Also, as the pattern is being matched each pattern of length m can have at most $m \times e$ possible mismatches.

After the search for all possible matches is finished a number of possible motifs are reported by the algorithm. In this stage Weeder uses a statistic constructed from the suffix tree to test the statistical significance of the found motif to sort the output.

Consensus

Consensus [21] is also a word based method which aligns the motif by scoring alignment matrices and testing for their statistical significance. While the first version of Consensus was dependent on the order of the input sequences supplied to the algorithm, the authors later fixed this problem and in the current version, to search for possible motifs one word of length k (motif length) is selected from each of the input sequences. Each word is then

turned into an alignment matrix. The alignment matrix that consists of one alignment has a 1 in the corresponding place of the base and 0 elsewhere. This process is then continued by adding all other not previously aligned words from all other sequences to each of these alignment matrices. So effectively each alignment matrix or each word is compared with all other words of length k . The comparison criteria is calculated from the formula:

$$\sum_{j=1}^L \sum_{i=1}^A f_{i,j} \ln \frac{p_i}{f_{i,j}}$$

where f_{ij} is the frequency that letter i occurs at position j . This formula, also known as relative entropy, is used to score all possible alignments. Each alignment matrix is updated by the best two-sequence alignments with the highest score calculated by the above formula. The algorithm continues until each sequence has contributed one or more (user specified parameter) words to the alignment. The p -value of the relative entropy statistic is calculated using a large deviation technique for statistical significance. Because enumerative methods cover the entire search space, they do not run the risk of getting stuck in a local optimum. On the other hand they may overlook some subtle patterns in the search space [14] and also since they look for all possible n -mers they are not suitable for larger motif lengths and sequences.

3.2.2 Alignment-Based Methods

Alignment based algorithms consist of probabilistic models that attempt to fit a motif model to the target sequence using a *background sequence model*. This type of model assumes that the entire sequence was created by a probabilistic model called the background model, but that the motif was created from another model called the foreground model or the motif model. Since the motif is statistically over-represented in the sequences, these two models can be identified.

In these algorithms the probability of a sequence being produced by the motif model given the background model is calculated. Different types of background models like random sequences or i -th order Markov chains are created and then compared with the sequence containing the target gene for over-representation. For discriminating between the background and foreground sequences, an optimization technique can be used to maximize the

likelihood of the observed data.

Motif Sampler [49] and AlignAce [25] use the Gibbs sampling optimization technique and MEME [5] uses the expectation maximization technique. DEME [37] is a discriminative motif discovery algorithm that uses a combination of local and global search to find the motif that best differentiates the positive and negative sequence sets provided as input. BAMBI [26] uses Bayesian inference and sequential Monte Carlo using a *Hidden Markov Model* (HMM) to model the occurrences of the motif. Motif sampler's Gibbs sampling search is very computationally expensive. MEME and other algorithms like DEME, which rely on background sequences, are very sensitive to the type of background sequence used and also require the user to procure a suitable background sequence otherwise they do not achieve very good results. Below some more popular methods of motif discovery using profile-based methods are analyzed.

MEME

This algorithm approximates maximum likelihood estimates of the parameters of a mixture model that has created the dataset. A finite mixture model assumes that observed data from the model comes from two or more groups with known distributional forms but different unknown parameters. *Expectation Maximization*(EM) is used to find maximum likelihood estimates of these unknown parameters. The algorithm's objective is to find those values of the parameters of the overall model that maximize the likelihood of the data. The mixture model does not actually model the dataset but instead it models a sequence of all overlapping subsequences of a given length. This is a good approximation if care is taken to ensure that the model does not predict two overlapping subsequences to be created by the motif model. This is done by enforcing a constraint on the estimated probabilities of two overlapping sequences being motif occurrences.

The mixture model consists of two components. One models the motif and the other background (non-motif) sequences. The motif model regards each position of the subsequence which is an occurrence of the motif to be generated by an independent random variable describing the multinomial trial. This multinomial distribution has parameters $f_i = \{f_{i1}, f_{i2}, f_{i3}, \dots, f_{im}\}$ where f_{ij} is the probability of letter a_j appearing in position i of motif of length m . These parameters are going to be estimated from the data.

The background model regards each subsequence of length W that is a non-

motif to be created as a series of W independent samples of a single background distribution with common parameters $f_0 = (f_{01}, f_{02}, \dots, f_{0L})$.

The model is assumed to be working in these lines: first the motif model with probability λ_1 or the background model with probability $\lambda_1 = 1 - \lambda_1$ is chosen. After this a sequence of length W is generated according to the probability governing the model chosen. The parameters for the model are:

- $\theta_1 = (f_1, f_2, \dots, f_W)$
- $\theta_2 = f_0$

where each f_i is a vector of letter frequencies of the motif model and f_0 is the vector for the letter frequencies for the background model.

The EM algorithm makes use of the concept of missing data. In this case the missing information is the group that a certain sample belongs to. The samples are represented by labels Z_{ij} which if equal to 1 mean that sample X_i belongs to group j . These labels are to be estimated along with θ (distribution parameters) and λ (mixing parameters).

After the parameters for the model are fitted to the data. The likelihood of a given pattern can be calculated from the mixture model. The authors of MEME report MEME's advantage over Gibbs Sampling based algorithms, which they believe to be the best at the time, is that it converges predictably and it also doesn't require the input sequences to be all classified as containing the motif.

Amadeus

The Amadeus [30] motif finder algorithm uses a series of filters or refinement stages to find the highest scoring motifs. The stages take a series of candidate motifs as input and try to improve on the scoring of these motifs. These newly scored motifs would then be submitted as inputs to the next stage. The stages get more computationally intensive as this filtering process progresses. This program also supplies an intuitive and simplistic user interface which reports on the motif's p -value, sequence logo, local localization score and some other statistical scoring schemes. The different stages of this algorithm are as follows:

Preprocessing considers and evaluates all possible k -mers where k is an input parameter to the algorithm by the user.

Mismatch adds some degenerate forms to the algorithm.

Merge combines similar motifs. These stages are repeated until no new highly scoring motif is added to the candidates. In the three first stages the motifs model is a consensus model.

Greedy transforms the candidate motifs into a PWM model and tries to optimize these PWMs with an EM algorithm which iteratively tries to find the best cutoff in the PWM that gives the best score. The instances in the target set that are above this threshold are used to refine the new PWM and the iteration continues until there is no improvement.

Post Process If the occurrences of two found motifs overlap more than 5% the lower scoring motif is removed.

Pair Analysis This stage is optional. In this stage co-occurring motifs are found and reported. The different scoring schemes are then combined into a single p-value using the Z-transform.

One of the other contributions in this paper is the comprehensive Metazoan benchmarking suite they compiled. This dataset holds promoter sequences from 1000 bp upstream to 200 bp downstream of the transcription start site. The authors report that, this range is often used in computational promoter analysis. The total sequence length of the target sets is 383 kbp on average which is considerably higher than previous benchmarking compilations.

BAMBI

BAMBI takes as input a series of sequences and an upper limit on the number of motifs embedded in these sequences. It argues that these motifs can be found by modeling the input sequences as an HMM. The hidden variables are the motif's length, logo, instances and location in the sequence.

The induction technique of these variables is Bayesian inference. In this technique first an initial value for the unknown parameters of the system is randomly chosen (prior). This initial value is represented as a distribution of the parameters rather than just a value. This distribution is then updated using the update method which incorporates newly observed data into the induction algorithm and hence conditions the "posterior" distribution with this new information. This process at each iteration gives a *posteriori* estimate of the parameters.

BAMBI differs from other algorithms that use this approach in choosing its observations from the sequence set. While some algorithms use bases or subsequences in each sequence as observations, BAMBI takes the entire sequence as an individual observation.

The additional computational cost resulting from this choice however is mitigated by the choice of a sequential Monte Carlo method. This approach at each iteration approximates the posterior distribution of the system parameters. Ideally, one would like to sample from the distribution itself in order to fit the parameters. But since usually for complex cases such as the said model, sampling is either impossible or computationally infeasible, these samples are taken from another distribution which is easily sampled using a technique called *importance sampling*.

As the computational complexity increases with the dimension of the state vector of the HMM, the algorithm is broken down in two stages:

- First the the sequential Monte Carlo method is used to determine if each sequence has a motif or not and obtain an estimate on the PWM.
- The first estimate of the PWM is then used for a sequence of binary hypothesis tests.

Breaking down the algorithm into these two stages enables it to use informative priors for its PWM, even from other motif finder algorithms.

3.2.3 Evolutionary Techniques

There are many examples of the use of evolutionary computation for motif discovery in the literature. These motif finders include: FMGA [31], MDGA [10], GAMI [12] and MOGAMOD [27].

FMGA

FMGA uses the same framework and operators for their Genetic Algorithm (GA) as defined in SAGA [33]. For individuals' representation FMGA uses a consensus based method and uses IUPAC characters as the motif. For its fitness function FMGA counts the number of occurrences of subsequences that match the consensus with some penalty for the mismatches summed over all input sequences. The genetic reproduction operators are as follows :

- **Mutation:** The individual is first turned into a PWM by considering all matched patterns in the input sequences. The columns containing 1 are unchanged but the others are randomly mutated. Every pattern creates two mutated parents to be supplied to the crossover operator.
- **Crossover:** A one point crossover technique is used which changes the left side of a random point in the first parent with the right side of a random point in the second parent and vice versa, producing two children. To encourage the more conserved patterns (i.e. with less permutations in each column) the operator adds predefined penalties to columns containing IUPAC symbols in the pattern. The child pattern with a better score overall is retained. The number of mismatches and the gap penalty of the crossover operator can be provided by the user. The authors report setting the correct values for these parameters was found to be difficult when trying to find longer motifs.
- **Rearrangement:** This operator is used to change the IUPAC symbols back to the most frequent nucleotide base (A, C, G, T) according to its weight matrix. The base with the highest number of occurrences is chosen. This operator is applied only when the predicted motif pattern is unchanged for a certain number of generations.

The authors reported that the algorithm produces better results than Motif Sampler and is faster than MEME. FMGA considers motifs as recurring patterns and looks for the best match to a random pattern it generated failing to take into account any statistical significance of the said pattern. Also the fact that the punishment value for the IUPAC symbols is an input parameter suggests that the algorithm needs refining for each dataset accordingly, which adds to the workload of the biologist.

MDGA

MDGA [10] evolves a list of integers as starting position of TF binding sites for the series of input sequences. Each sequence is assigned a random starting position and the length of the motif is fixed by an input parameter. So the representation of each individual is the starting position of the motif it represents in each of the input sequences. This forces the restricting assumption that each sequence must contain the motif. The fitness function is the

entropy measure of the pattern. The entropy formula is the same formula known as relative entropy :

$$IC = \sum f_b \log_2 \frac{f_b}{p_b}$$

where f_b is the frequency of the nucleotide b in the column of the found instances of the motif and p_b is the frequency of this nucleotide in the background sequences.

Single-point and double-point crossover and bitwise mutation are used for reproduction. The replacement strategy in the generational GA is as follows: in each generation 100 individuals are used to create 50 new individuals which are added to the parents, and then the worst one-third of these 150 individuals are eliminated. The authors reported that the algorithm produced better results than Motif Sampler and also was faster than Gibbs Sampler and BioProspector.

The authors acknowledge the fact that the assumptions made about the necessity of a motif occurrence in each sequence is highly restrictive. Also the fact that the columns of the prospective motifs are considered to be statistically independent should be highlighted, as this is restrictive to the motif model.

GAMI

GAMI [12] uses a similar approach to FMGA, evolving motif patterns and matching as many subsequences base matches to the pattern as possible. The overall fitness is the maximum number of base matches for a pattern throughout the sequences. GAMI tries to find motif patterns in the space of all possible N -mers, where N is the length of the motif, compared to looking at all possible overlapping subsequences of a given length. The authors argue that this considerably reduces the search space by making the algorithm dependable on the motif length rather than sequence length.

GAMI expects at least one motif per sequence. It also uses the punishment used in FMGA to encourage the occurrence of A,C,G,T rather than IUPAC characters in any column. The mutation operator truncates one end of the motif and adds a random base to the opposite side allowing the motif to “glide” along the sequence.

In a later paper [11] the “Information Content” fitness function was used: this was reported not to be effective since it did not require the motif to

match the sequence completely and could have many degenerate forms.

MOGAMOD

Multi-objective GA approaches to motif discovery are rare, but MOGAMOD [27] is one such example. MOGAMOD builds on the idea of MDGA and represents the individuals as starting positions of binding sites in each sequence with two main differences.

Firstly, unlike MDGA there is no requirement for all sequences to contain the motif: the occurrence of motifs in each sequence is determined by a weight coupled with the starting position, which when less than a given threshold does not regard the sequence as containing the motif. Secondly, the length of the motif is evolved with the individual.

The objectives used in MOGAMOD are as follows :

- **Similarity:** The gene representation used here, encodes different starting positions of the motifs in the sequences and their length. These positions may point to completely different patterns that could not possibly constitute a motif. This objective, scores the patterns that are similar to one another. In order to do so, firstly all the patterns currently found by the individual are converted into a PWM. This PWM is then scored by averaging the highest frequency of a base per column. The frequencies are numbers between 0 and 1 so the average would also fall in this range and the closer it is to 1 the more conserved and the more similar the patterns are.
- **Support:** This objective got its name from a data mining concept. Since the motif may exist in many or no sequences in the input sequences set, this objective indicates the number of sequences in which the motif was found. The more sequences the motif was found in the better the quality of the found motif.
- **Motif Length:** This objective is maximized since the program looks for the longest motif it could find.

MOGAMOD uses an arithmetic crossover and three mutation operators. The crossover operator is a single point crossover which takes a parameter that fixes the point to be used as the pivot. This point is a percentage of the

length of the individual and is changed as more generations are processed. The mutation operators are as follows :

- Shift the starting location of the found motif to the left.
- Shift the starting location of the found motif to the right.
- Random change: A random number is added or subtracted from either length, weight or the starting location of the motif.

Although MOGAMOD improves on the GA setup of MDGA and FMGA or GAMI, the current representation adds the overhead of first finding patterns with acceptable similarity. This problem can be easily solved by a better choice of representation which will be discussed in Chapter 4.

MOGAMOD showed very strong results on the 3 datasets chosen from the Tompa [50] benchmarking suite. The reason why the specific datasets were picked out of the many other datasets available in that benchmark is not mentioned. The algorithm was shown to be able to match the results from MEME, AlignAce and Weeder and improve them on the motif length objective. The algorithm was also reported to be faster than MEME.

Chapter 4

Side Effect Machines

In this chapter the representation used for a motif in our motif discovery tool is elaborated and some early experiments done with the setup for DNA classification are reported.

4.1 Definition

Side Effect Machines (SEMs) were introduced in [3]. The idea behind the SEM is associating useful information with each state so that when a sequence is run through the machine a numerical feature set is extracted from the run. An example of a side effect is state visit count, i.e the number of times the state was visited when the sequence was run through the machine. This side effect makes the SEM model a perfect model for feature extraction from a variable length sequence of events. When the series of events are parsed through the SEM (each state representing an event) the state visit counts produce a count vector when the parsing is finished. The key factor here is that the count vector's length is not dependent on the length of the series of events but on the number of states of the SEM. This means SEMs can easily model variable length sequences.

We further introduce SEMs by following through an application of them in *Bioinformatics*. A complication when using computational intelligence in Bioinformatics is the use of feature vectors by most techniques: in the case of variable length sequences, these are not easy to extract. A *Side Effect Machine* (SEM) can be used as a feature inducer of the DNA sequence.

In [2] the authors used the count of the number of visits to each state as the side effect for the SEM. We use the same strategy in this example. Extraction of the features works as follows.

The SEM starts at state zero. A DNA sequence is fed to the SEM one base at a time, making transitions to other states according to the base parsed and the SEM's transition matrix. This process continues until all bases in the sequence have been parsed. The number of times each state was visited during the process creates a feature vector of length equal to the number of states. We use the same approach to find motifs in sequences. An example of a SEM and its transition matrix is shown in Figure 4.1. The feature vector produced from parsing a sample sequence through the SEM in Figure 4.1 is shown in Table 4.1.

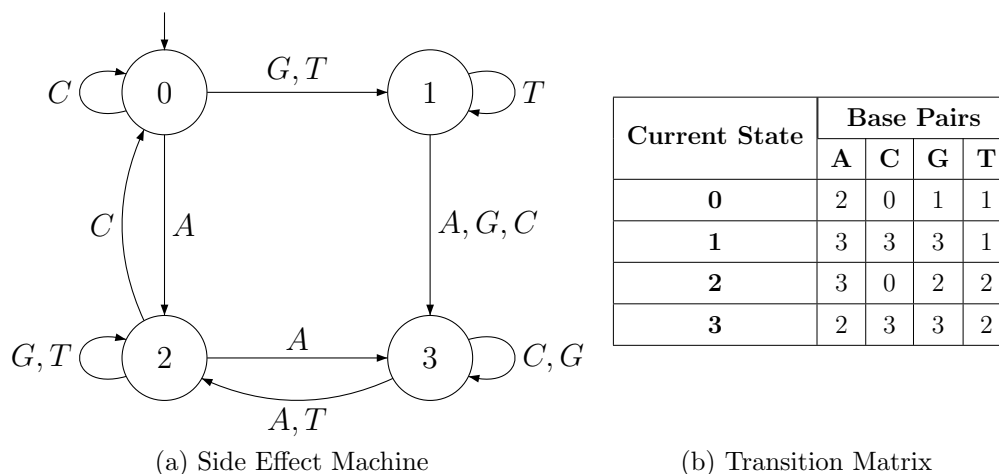


Figure 4.1: SEM and Transition Matrix

4.2 SEMs and DNA Classification

To meet the demand of newer sequencing technologies that provide sequences of higher length and quantity, sequence alignment and classification algorithms need to evolve. A plethora of multiple sequence alignment and classification algorithms has been developed to meet this need. A very good review and comparison of these algorithms can be found in [24, 28].

Previously SEMs were shown to be useful in classifying DNA Sequences [2, 4].

Table 4.1: Sample Count Vectors

Sample Sequences	Feature Vector			
	0	1	2	3
CCGTAC	3	2	0	2
CAGTTG	2	0	5	0
AAAACG	1	0	2	4
GGGGGG	1	1	0	5

The authors reported that the SEM was successful in classifying the DNA sequence and also, that it was able to pick up on complex patterns in the sequence. In order to analyze the inner workings of the SEM and how it is able to come up with the classification by detecting these complex patterns, some experimentation is done on the classification method reported in the said papers. We construct synthetic datasets with different properties and then run the sequences through SEMs that are evolved with a GA. The approach is elaborated below.

4.2.1 Dataset For Classification

We are going to use SEMs for DNA classification. For this purpose, a synthetic dataset containing a set of sequences is created to supply the sequences for classification.

The datasets created are going to test if the SEM is able to pick up on the motif if it exists inside the sequences. The dataset can be designed having a certain number of the same motif embedded in the sequences. The embedded motif chosen is HNF-1 with the consensus sequence “TATTGTTTATT”. The dataset is created so that half of the sequences contain the specific number of the motif and half do not contain any.

The dataset is first created by creating sequences of the required length assuming a uniform distribution over the bases (all have the same probability of occurrence of 0.25). The embedded motif is then replaced with a subsequence of the same size in a random position. Care is taken not to overlap an already inserted motif.

In this experiment, the embedded motif is inserted in the sequences with one *mismatch* to simulate biological datasets. The length of the sequences

and the number of embedded motifs are determined and explained in each experiment. The experiment's input is a set of sequences created from either of the two possible datasets. The input sequences are divided into two groups for cross validation purposes:

- *Exemplars* which are sequences that are chosen to train the SEMs in the training phase.
- *Evaluators* which are sequences used to test the classification performance of the SEMs.

4.2.2 Representation

The GA uses SEMs as individuals which are represented using their transition matrices. These SEMs are initially generated by randomly creating transition matrices of size $m \times 4$ that is supplied as a parameter to the algorithm. The transition matrix has 4 rows per nucleotide and m columns per state.

4.2.3 Reproduction

The reproduction operators are one-point crossover and mutation. Crossover swaps two rows of the transition matrix in the two selected parents. These rows are randomly chosen. Mutation changes a row in the transition matrix by changing the transitions to random states in the SEM.

An example of two parent SEMs are shown in Figure 4.2. The resulting offsprings are shown in Figure 4.3.

4.2.4 Fitness Function

In the evaluation phase each individual (SEM) parses each exemplar sequence in the dataset as was discussed in section 4.1. As the sequence is parsed through the SEM each count vector is stored along with the sequence it was extracted from. *K-Nearest Neighbor* (KNN) is then used to classify the set of exemplar count vectors.

The KNN algorithm [17] is a non-parametric machine learning technique that classifies objects according to their nearest labeled data objects using a major vote technique. The assumption is that each data object is in the same class as the majority of its K neighboring data objects. In a non-parametric

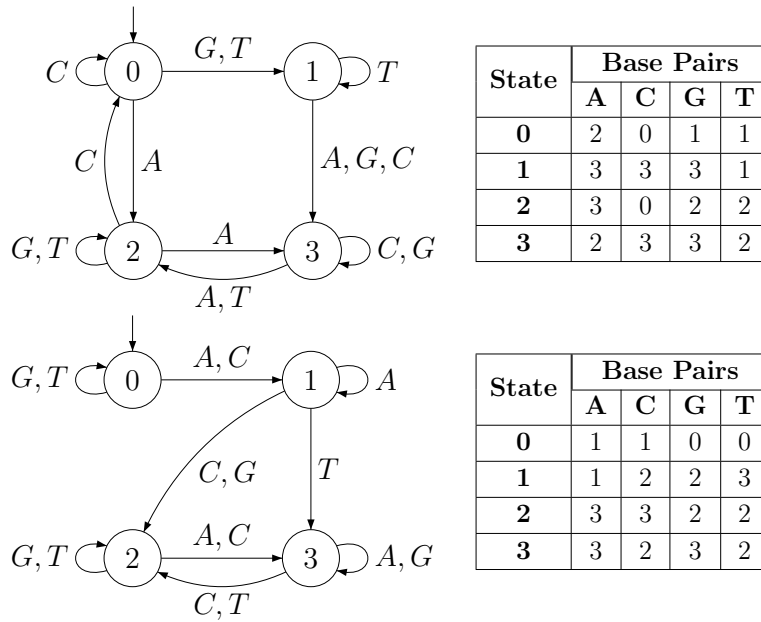


Figure 4.2: Parents Before One-point Crossover

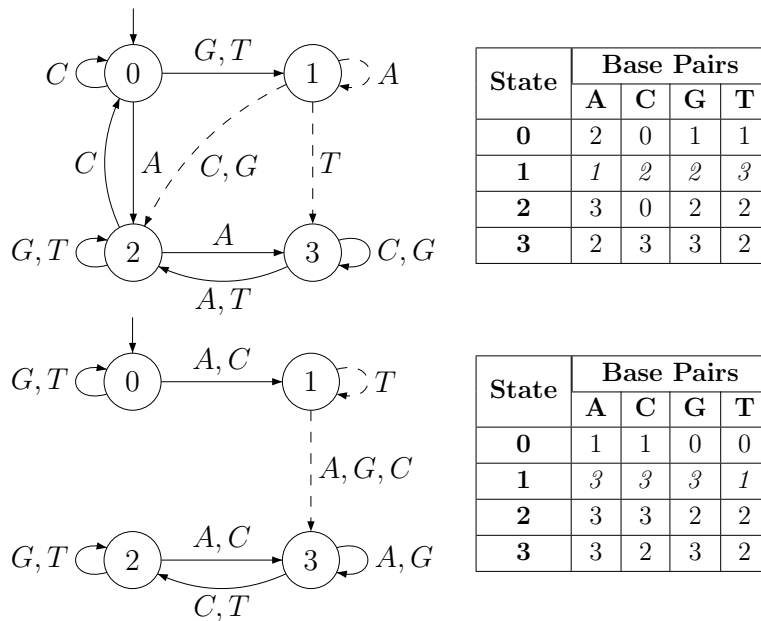


Figure 4.3: Offsprings After One-point Crossover

approach the model does not fit any parameters to the data using a training set. Therefore, KNN being non-parametric and classifying according to known instances can be classified as an instance based, lazy classifying algorithm.

In our experiment, the data objects are count vectors extracted from each DNA sequence. The KNN algorithm labels each count vector to be of either class according to its neighboring count vectors. The distance measure used is *Euclidean Distance* which measures the smallest distance using a straight line between two points in space.

In order to test the classification accuracy the *Rand Index* estimate[15], described below, is used to test two sets' similarity. Given a set of n elements and two partitions of the set S to compare, the following is defined:

- The number of pairs of elements in S that are in the same set in X and in the same set in Y .
- The number of pairs of elements in S that are in different sets in X and in different sets in Y .
- The number of pairs of elements in S that are in the same set in X and in different sets in Y .
- The number of pairs of elements in S that are in different sets in X and in the same set in Y .

The Rand index, R , is a number from the range $[0, 1]$. This number gives a measure of similarity between two classified sets. In this study the two sets X and Y are the known classification of the sequences and the predicted classification of sequences respectively. Values close to 0 mean the two sets don't agree at all and values close to 1 mean that the two sets are exactly the same. This estimate works by counting the number of elements correctly classified and number of elements incorrectly classified. As the number of misclassified elements grows the value approaches 0.

Rand index is used as the fitness function in this study: if an SEM is able to achieve a fitness value of 1 then the ideal individual and the solution to the classification is found.

4.2.5 Experiments and Analysis

This experiment is to test if the SEM is able pick up on the motif when the motif exists in the sequence. The synthetic datasets created contain 1000 sequences each from 1000 to 2000 base pairs long. In order to find how existence of the motif could change the sequence base composition, five different datasets with 10 to 50 embedded motifs were created. The exemplar and evaluator sequences are picked at random from the entire dataset in the specific experiment. The parameters used for the experiment are shown in Table 4.2.

Table 4.2: Relative Entropy vs Entropy Experiment Parameters

Parameter	Value
Number of Runs	30
Number of Generations	50
Mutation Rate	30%
Crossover	70%
Population	100
Selection Method	Tournament Selection (Size 3)
Exemplars	25
Evaluators	15
Elites	2
Number of States	2

The experiments on these 5 datasets revealed that even a two-state SEM is easily evolved in the first few generations when the embedded motif occurs in the sequence set more than 30 times. When the number of occurrences falls below this threshold the performance deteriorates and falls to 95% with 20 occurrences and 77% with 10 occurrences.

The main reason for the GA's success in finding the best individual so fast when the number of occurrences is high is that with a high number of occurrences of the motif the GC-content of the entire sequence set is completely effected by the motif. Table 4.3 has the different GC-content percentages for each dataset.

Table 4.3: Percentage of Bases

Number of occurrences	A	C	G	T
10 Embedded Motifs	0.24	0.23	0.24	0.27
20 Embedded Motifs	0.24	0.21	0.24	0.29
30 Embedded Motifs	0.24	0.20	0.23	0.31
40 Embedded Motifs	0.24	0.18	0.23	0.33
50 Embedded Motifs	0.24	0.17	0.22	0.34

Table 4.3 shows that the GC-content could be considerably affected when the number of motifs embedded is high especially if the motif is rich in certain bases which is the case with the embedded motif here. The table shows considerable change in the GC-content down to 10 embedded motifs. The rather low performance in this dataset then, is due to the small difference made to base composition.

The SEM then would only need to count the number of G and C bases in each sequence to guess if the motif is embedded in the sequence or not. The two-state solution evolved is shown in Figure 4.4.

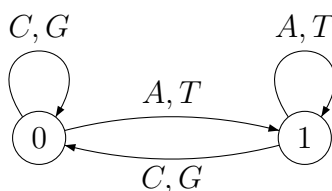


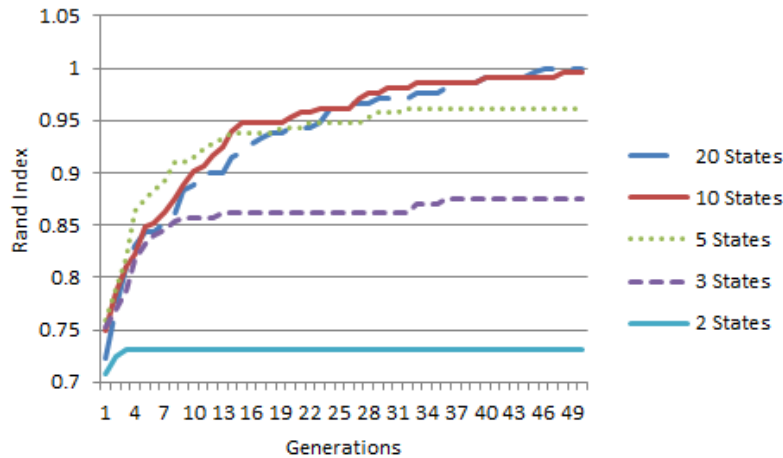
Figure 4.4: A Simple Two-State SEM

In the next experiment the algorithm is tested against a dataset with only one embedded motif. The previous experiment shows that the algorithm would no longer be able to detect the motif with base composition analysis since in this experiment with only one added motif, the base composition remains roughly the same and indiscernible to the SEM count vectors.

In the dataset created for this experiment, sequences of length 100 are created with only 1 occurrence of the HNF-1 motif (with an added mismatch). The parameters used for this experiment are the same as the previous experiment

(Table 4.2) with the difference that the state count is variable. The results are shown in Figure 4.5.

Figure 4.5: DNA Classification with Variable State Count



As shown in Figure 4.5, the algorithm was able to successfully classify the sequences with one motif embedded in them. The motif does not change the base composition considerably so it can be deduced that the algorithm has picked up on the pattern embedded in those sequences.

The relation between the state count and the performance is also worth noting. The two-state SEM as shown does not have good classification accuracy or a good evolution since there is not much to evolve with only two states. As the number of states are increased the algorithm is able to produce more complex patterns and after 10 states the performance is excellent.

Considering the motif was found in the sequences, how can one extract this motif from the SEM? The extraction of the motif from the SEM proves to be an equally hard problem. This can be seen in Figure 4.6: what pattern does this SEM find? The figure shows the best individual evolved in the run with three-state SEMs. Although this SEM does not have the accuracy of a 20-state SEM classifier, it shows the difficulty of motif extraction from even three states of the SEM.

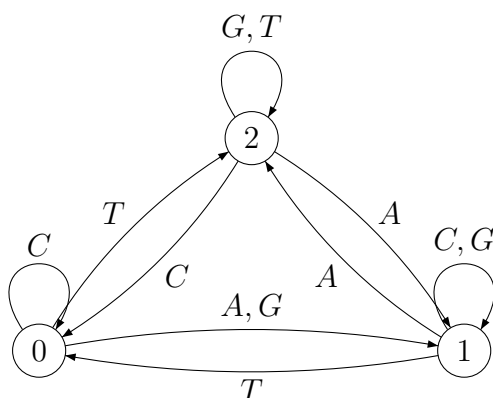


Figure 4.6: Detector of HNF-1 with 87% accuracy

4.3 Restricted SEM for Pattern Recognition

In the previous section, the experiments showed that although the SEM found the motif (according to the high classification score), the inner workings of a SEM of even 3 states was already sufficiently complex that extracting the motif was very difficult. To circumvent this problem we restricted the SEM to only develop patterns that could later be easily extracted. Functions like loops and jumps to states ahead, although helpful for finding the motif, are not required for pattern discovery since both can be simulated by a SEM with a higher number of states but without those functions.

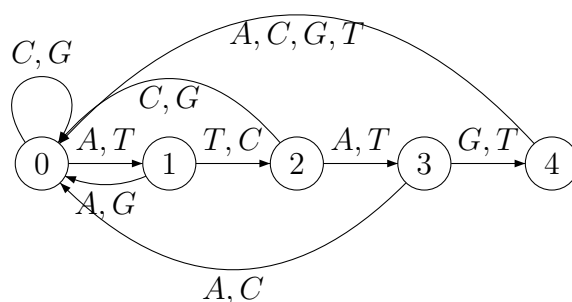


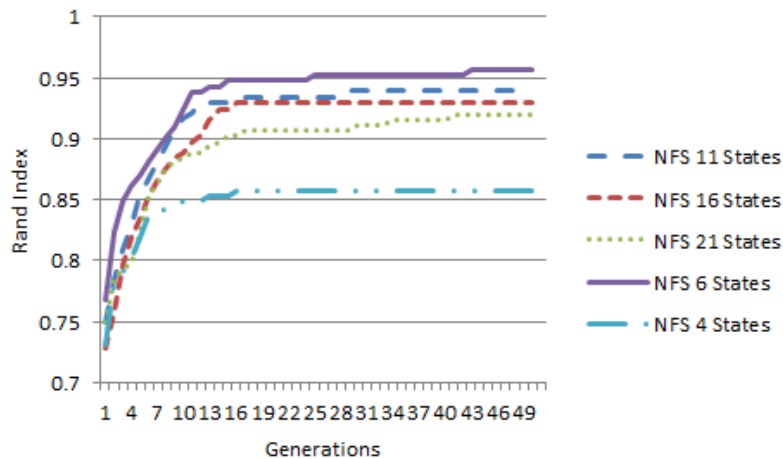
Figure 4.7: Next or First State Side Effect Machine

A *restricted-transition* SEM is one in which certain transitions are disallowed. In our work, we limit the transitions of a n -state SEM as follows:

for each state i , $0 \leq i < n$, the only transitions allowed from state i are to state 0 and to state $i + 1$ (if it exists). We call an SEM designed with these restrictions a “*Next or First State Side Effect Machine*” or NFS-SEM. An example of a NFS-SEM is shown in Figure 4.7. The restriction allows the NFS-SEM to be used as a pattern recognizer that makes transitions forwards while on the pattern and that goes back to the first state to start over when the sequence diverts from the pattern. NFS-SEMs have the last state set as a “Pattern Found” state, meaning that if this state is visited then a pattern has been found in the sequence. In further experiments NFS-SEMs were able to gain acceptable classification accuracy for the same DNA classification problem normal SEMs were applied to.

It may be assumed that since the loops and jumps in the SEM are now not allowed, the classification would have poor results. To further analyze this, the same dataset from the previous section with only one motif embedded with one mismatch is also used this time with an NFS-SEM to see how the classification accuracy changes. The results of this experiment are shown in Figure 4.8.

Figure 4.8: DNA Classification with Variable State Count



As the Figure 4.8 shows, although the 100% accuracy rate is not achieved, the NFS-SEM is able to obtain up to 95% accuracy which seems reasonable given the restriction.

Chapter 5

Genetic Algorithm Design

This chapter explains the details of the motif discovery tool created in this work. The representation, reproduction operators, fitness and strategies used are detailed.

5.1 Representation

There are notable differences in choosing SEMs for representing the motif instead of consensus based methods like GAMI, YMF, Weeder, etc. SEMs have the same capability as consensus methods to show degeneracy by allowing more than one transition from each state. They don't need to use IUPAC characters to represent the motifs, nor do they need to set the number of mismatches because intrinsically they handle the degeneracy by state transitions. Another advantage is the incorporation of a fast method to parse the sequence. In comparison, consensus based methods parse the string by moving a motif-length window along the sequence and calculating the mismatches, requiring more time than just parsing the string through a SEM. A final advantage is that after parsing, SEMs output a feature vector (state visit counts) for each sequence parsed which is usable as a problem specific heuristic.

5.2 Sexual Reproduction

Since the SEM used is a NFS-SEM, a general crossover or mutation operator may render an individual invalid by exchanging states at non-corresponding

indexes or mutating a state transition to point to an invalid state. Consequently, special reproduction operators are needed. Two types of crossover and three types of mutation are considered.

5.2.1 Crossover

Crossover State At Index This operator is the normal one point crossover in which a random state is chosen in one of the parents and all the transitions are swapped between the two at the same state. Figure 5.1 shows an example of the two parents before this crossover and Figure 5.2 shows the corresponding result after crossover. The dotted lines represent the transitions that are exchanged in the parents. As seen in the figure the index 2 was chosen for crossover and all transitions to states 3 and 0 were swapped.

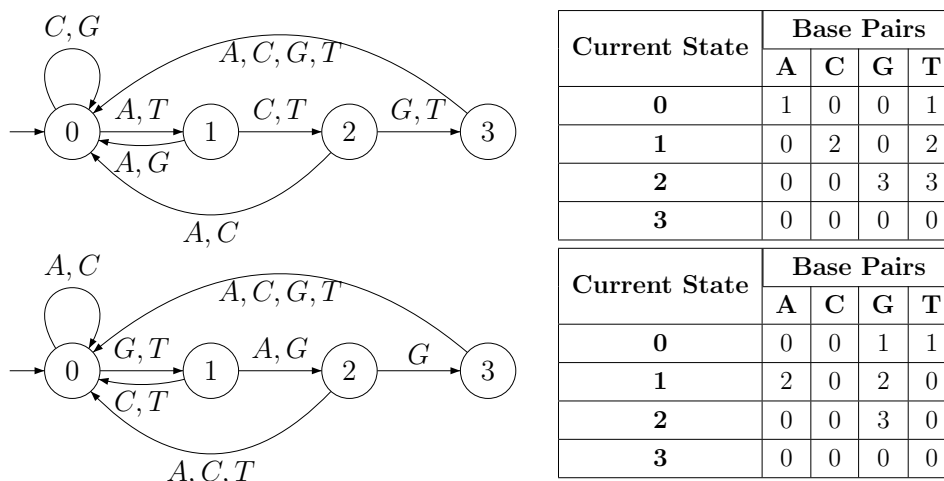


Figure 5.1: Parent SEMs before the State At Index Crossover

Crossover Genetic Material In this crossover two random states are picked. All transitions between these two points are swapped between the two parents. By applying this operator, individuals that have correctly determined the conserved part of the motif can pass the genetic material to individuals that have correctly determined the less conserved parts. Figure 5.3 shows the result of applying this crossover to the same parents from

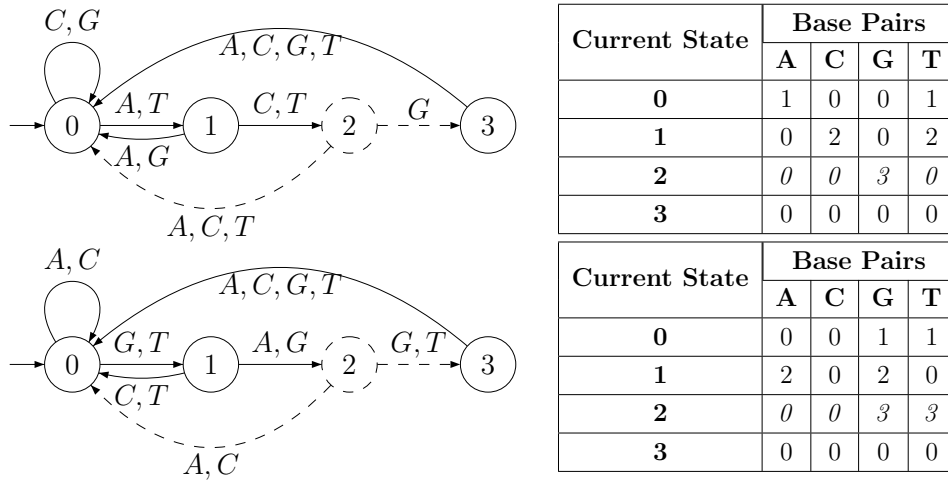


Figure 5.2: Parent SEMs after the State At Index Crossover

Figure 5.1. In the figure states 1 and 2 are chosen for substitution and all transitions between them are swapped.

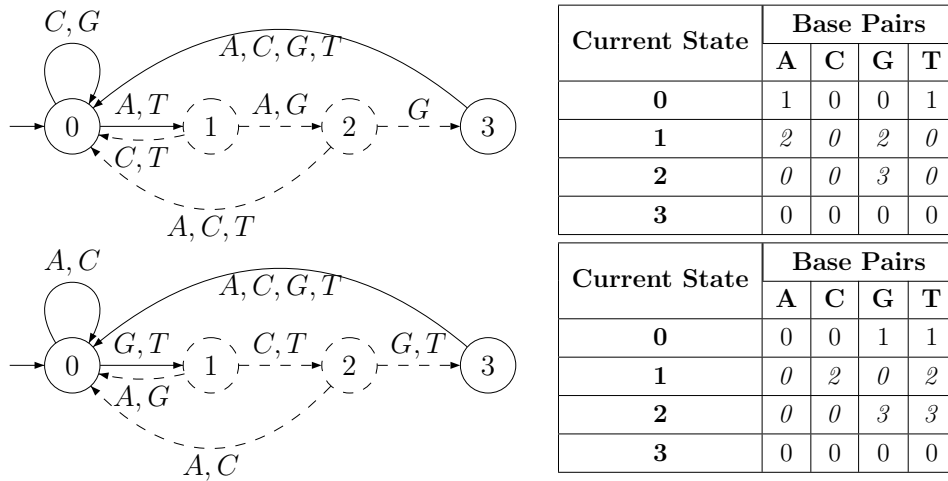


Figure 5.3: Parent SEMs after Genetic Material Crossover

5.2.2 Mutation

Random Transition In this type of mutation a random state is picked and for all its state transitions a coin is flipped to change the transition to

the next state or the first state. The individual is then checked and the process repeated if all transitions were changed to the first state at a certain index, rendering an invalid individual. Figure 5.4 and 5.5 show this operator in action.

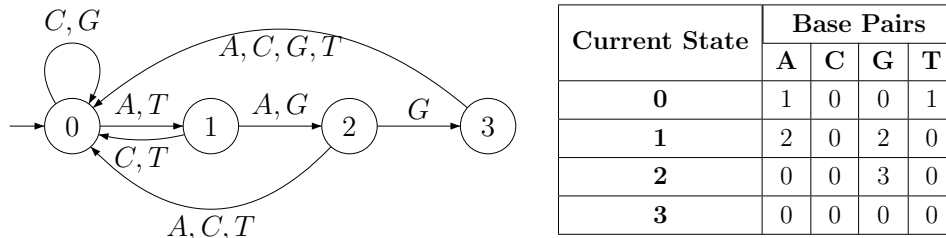


Figure 5.4: Parent Chosen for Mutation

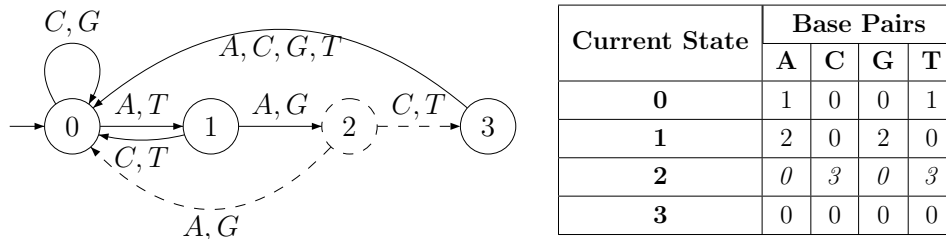


Figure 5.5: Offspring After Random Transition Mutation

Frontal Decay In this type of mutation, the first state is deleted and a new state with random transitions is added to the penultimate state (the last state is the “Pattern Found” state and does not take part in reproduction). Figure 5.6 shows the result of applying this operator to an individual. The dashed lines represent the transitions mutated.

Posterior Decay This operator is the opposite of the previous one. The NFS-SEM would lose its penultimate state and receive a random starting state with random transitions (obeying any required restrictions). Figure 5.7 shows the result of applying this operator to an individual. The dashed lines represent the transitions mutated.

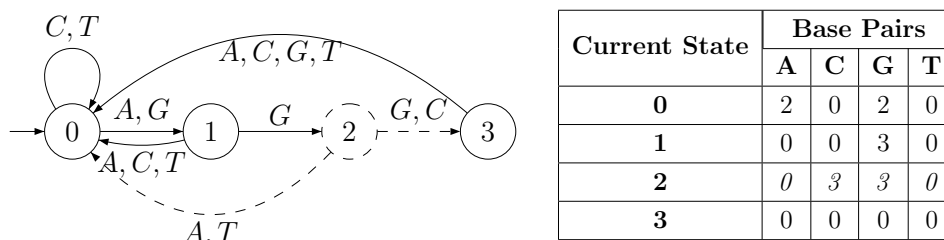


Figure 5.6: Offspring After Frontal Decay Mutation

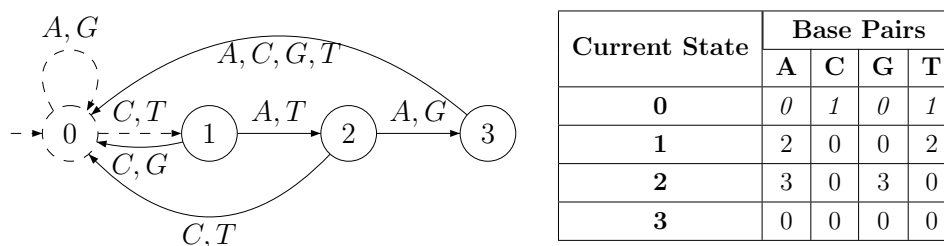


Figure 5.7: Offspring After Posteriorl Decay Mutation

5.3 Multi-Objective Fitness

One characteristic of a motif is its number of occurrences. In our implementation if only this objective is taken into account then the GA would look for an NFS-SEM that has the highest number of visits for its last state. There is an NFS-SEM that would have the maximum number of state visits possible for any sequence fed to it and that is one that for all states the transition to the next state happens for any base. This is shown in Figure 5.8. This easily achieves the maximum number of visits to the last state objective but clearly the NFS-SEM would merely represent a motif pattern of all possible n -mers and is never a true solution to the problem. Consequently other objectives of the problem should be considered as well. Our main goal then is to search for a motif that allows for degenerate forms but is conserved as well. We now suggest and describe several objective functions.

5.3.1 Objectives

Last State Visit Count This objective counts the number of times a certain pattern was found which is the same as the number of times the individual's NFS-SEM parsed the string and was able to reach its last state.

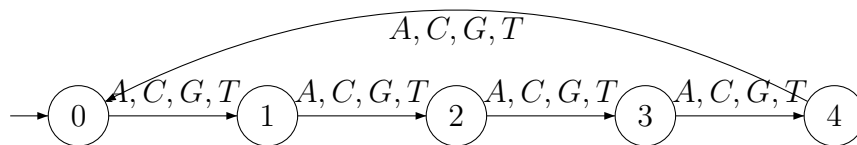


Figure 5.8: NFS-SEM with all transitions going to the next state

This objective is maximized. Since the NSF-SEM is already storing visit counts for each state, this objective’s value is the last index of the extracted feature vector.

Entropy This objective shows the amount of uncertainty in each position of the motif. The more conserved a motif is, the lower the entropy. The formula used for entropy uses the same concept as Shannon’s entropy used in Information Theory [47]. The entropy for a motif pattern M is calculated as:

$$E(X) = - \sum_{i=1}^m \sum_{j=1}^4 P(x_{ij}) \log_2 P(x_{ij})$$

where m is the length of pattern M and X is the probability of occurrence matrix with x_{ij} the probability that base j happens in the position i of the motif. The entropy is calculated from the individual’s PWM by first calculating the probability of occurrence of bases at all locations of the motif and then using the above formula to calculate the summation of all bases in each location. The resulting entropies of all columns are then added together for the motif’s entropy value.

The highest value for entropy in each column is two, which occurs when all pairs have an equal probability of occurring. The absolute value of entropy is minimized in this problem. It is also in opposition with the first objective. The “Last State Visit Count” objective tends to favor very degenerate SEMs to be able to get to the last state and get better fitness, but such an SEM would have a very high entropy value and is not going to receive a good final score.

Clustering SEMs were shown to be successful at classifying DNA sequences in both this study and earlier studies [2, 4]. It was also used to find motifs in [32] coupled with an evolutionary algorithm. This prompted the idea of using them as an optimization objective in this problem.

In the context of discriminative motif discovery this objective can be turned into a classification objective. For the current non-discriminative problem then clustering is adopted as another objective. Feature vectors extracted from each string in the sequence are going to be fed to the k -means clustering algorithm. The algorithms would then cluster the instances to two clusters, aiming to minimize the intra cluster error and maximize the inter cluster distance.

Likelihood This objective relates to the fact that motifs are statistically over-represented in the dataset being analyzed. Firstly, a probabilistic background model for the input sequences is assumed. Then, according to this background model, the input sequence is fitted to an n -order Markov model which would then be used to calculate the likelihood of a pattern to be produced by the background model. Due to statistical over-representation, the aim here is to minimize the likelihood of the pattern given the background model.

5.3.2 Scoring Method

Individuals in a multi-objective problem have a vector of values as their fitness, so comparing them is not straight forward. Pareto ranking ranks the individuals by a domination criterion and is known to be susceptible to outliers [38]. In our work the normalized sum rank [7] approach was taken. For all objectives of the problem, the population is sorted by that objective and individuals are ranked according to their value in that objective. This result is a rank vector which is then summed to be the individual's fitness. This converts a multi-objective problem to a single objective problem and there is no need to assign weights to different objectives.

Although weighted ranks could be used to weight the importance of certain objectives, we do not use this approach here. Also, a normalized ranking scheme was chosen since when ranking individuals in the population some objectives could lead to having differences in ranges of values in ranks due to identical and repeated values of the individuals for that objective.

The problem with summed rank is its relativity to each generation. The score of an individual is dependent only on individuals of its own generation. This can result in the best individual of the run possibly being overwritten by a worse individual in generations to come. To mitigate this problem we use an archive to hold all the undominated best individuals of all generations

of the run. A best individual of each generation is added to the archive only if it is undominated by all the individuals in the archive. The archive size does not have a limit in our experiments.

5.4 Algorithm Outline and the Archive Set

The algorithm starts by creating an initial population of NFS-SEMs as transition matrices which also represent a motif. An individual also holds a PWM that corresponds to the occurrences of the motif updated online as the sequences are parsed. The update happens every time the NFS-SEM hits the last state (the “Found Motif” state). After each evaluation the best individual of the generation is analyzed to determine if it should be put into an archive set. The requirements are as follows:

- Not Pareto Dominated: There should not be any other individual in the archive that Pareto dominates the individual to be added.
- PWM Divergence: To encourage diversity in the archive, the PWM of the individual to be added is compared to all of the individuals in the archive. This measure calculates the Euclidean distance of corresponding columns in PWMs. Only individuals with an average Euclidean distance greater than 0.25 (number set after extensive experiments) are allowed in the archive otherwise the individual is deemed to be finding a very similar motif and is rejected.
- Degenerate Consensus Forms: The consensus form of the individuals in the archive and the individual to be added are taken from the position weight matrices by picking the most probable base to appear in each column. The candidate individual’s consensus is compared against all individuals in the archive. If any of the distances are zero or less than $\lceil \frac{PatternLength}{4} \rceil$, the new individual is rejected.

At the end of each run the contents of the archive are the found motifs in the dataset.

Chapter 6

Datasets

In this chapter the datasets used to test the performance of our tool and how they were curated or created are discussed.

Most motif finding algorithms introduced in the literature use datasets gathered from experimental results as in data from ChIP-chip experiments (biological experiments which experimentally determine protein binding sites during protein-DNA interactions) or by curating a set of co-regulated genes. Although choosing different datasets will prevent different algorithms from bias against a certain type of dataset, it may also lead to confusion from the biological community regarding the exact difference in performance and accuracy of motif finding algorithms. As a result, benchmarking datasets were created.

In our work we created a synthetic dataset to simulate a variation of the known Challenge problem [35]. We also use current available benchmarking datasets.

6.1 Synthetic Dataset

The synthetic dataset designed for testing of our work is a variant of the well-known Challenge problem [35] for non-discriminative motif finding algorithms. A standard synthetic dataset in this field would have a consensus motif in all of the sequences and the distribution of the bases in other parts not containing the motif is uniform. We introduce more complexity by only adding the motif in half of the sequences and adding not the consensus but various forms of a degenerate motif to the sequences. The original Challenge

problem dataset includes 20 sequences, each with a number of mismatches in the occurrences of the motif. The sequence length and the algorithm is tested on different lengths for the sequence up to a maximum length where they fail to find it. The synthetic dataset created here aims to simulate biological occurrences of a motif in a sequence or series of sequences. Therefore, instead of adding mismatches to the consensus sequence, a position weight matrix of a known motif is used as instances of the motif in the sequences. In real applications of this problem not all the sequences hold the motif, thus we also added negative sequences to the dataset. The synthetic dataset holds 40 unlabeled sequences consisting of 20 positive and 20 negative sequences. The positive sequences are created using the HNF-1 motif's PWM. First the PWM of the HNF-1 motif is analyzed for the probability of occurrence of each base in each location. Then two degenerate motifs are randomly created according to the calculated probabilities and inserted into a sequence of the desired size with randomly created base pairs. The positions of motif insertion are checked to avoid motifs overlapping each other.

6.2 Tompa et al's Dataset

The authors in [50] introduced the first prominent benchmarking dataset. It consisted of 56 datasets on a set of organisms. The authors also compared a series of motif finding algorithms. The comparison data of all these algorithms are easily available which makes it possible to compare our algorithm with other algorithms using the dataset, as well as MOGAMOD [27], which has been tested on this dataset.

6.3 Metazoan Compendium

The authors in [30] curated a set of co-regulated genes from high throughput experimental techniques in the literature. The sequences needed for using this data set were extracted from the Ensembl database using a sequence retriever tool [39]. The promoter sequences were extracted from 1000 bps upstream to 200 bps downstream of the transcription start site. This range as the authors report is mostly used in promoter sequence analysis. The total sequence length of the sequences are 383 kbp on average which is considerably larger than most other datasets to date. Their dataset contains Gene-IDs

from the Ensembl Database [44] which have been known to contain binding sites of the same transcription factor.

The different datasets in this benchmarking suite are named after the first transcription factor that they contain followed by the author of the paper the dataset was extracted from. For example the CREB_Zhang shows that the dataset contains the “CREB” TF and the author’s name follows.

Chapter 7

Results

This chapter holds the results of the execution of the algorithm on the datasets discussed in the previous chapter. The results are reported and the algorithm's performance is analyzed.

7.1 Synthetic Dataset

One of the design decisions of this program that differentiates it from others (e.g. FMGA, MDGA, GAMI and MOGAMOD) is the use of SEMs to represent the individuals. For all of these algorithms, performance suffers when the sequence lengths and number of sequences change. We have devised an experiment to test the effect of sequence length on our program. 10 datasets were created using sequences of length 100 to 1000 each with the same properties as described in 6.1 and having the HNF-1 motif (TATTGTTTATT) embedded, including degenerate forms.

The parameters used in this experiment are shown in Table 7.1. For this experiment only the first two objectives mentioned in section 5.3.1 were used. The performance measures chosen are the PWM divergence and the edit distance to the consensus form of the embedded motif. The output of each run consists of all the motifs found in the archive set. For each individual, the edit distance and PWM divergence to the embedded motif was calculated and averaged for all 30 runs. Results are shown in Figure 7.1 and 7.2.

For all the runs of different lengths the motif was found in the archive with

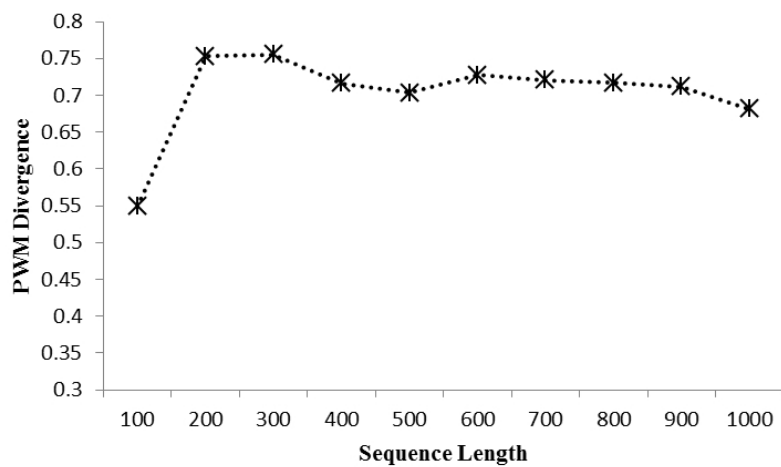


Figure 7.1: Average PWM divergence (30 runs)

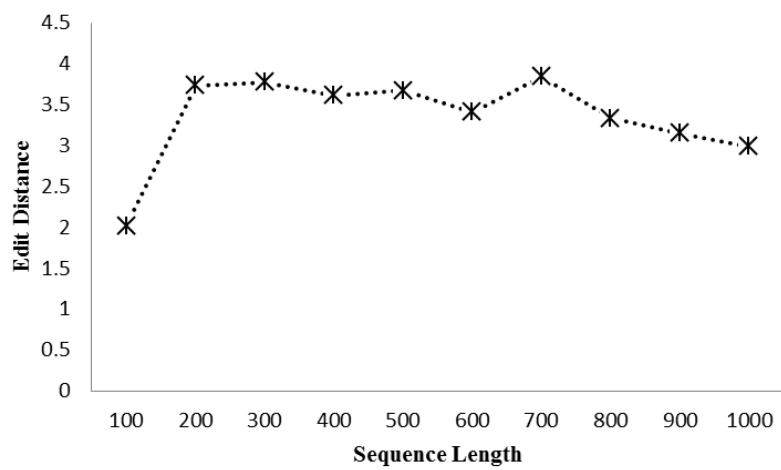


Figure 7.2: Average Edit Distance (30 runs)

Table 7.1: Parameters for Run on Synthetic Dataset

Parameter	Value
Number of Runs	30
Number of Generations	100
Mutation Rate	30%
Crossover	70%
Population	2000
Selection Method	Tournament Selection (Size 3)
Background Model	1st-order Markov Model
Objectives	Found Motif Count + Entropy
Motif Length	11

the exception of three datasets consisting of sequences of lengths 300, 400 and 500. In each of these three datasets a motif with an edit distance of 1 was the best motif found. An example of the best motif found is “TATTGTTTATA” which only differs from the target motif in the last base. By comparing the motif to the PWM of the target motif we realized that the motif found was actually not a mismatch but a degenerate form of the target motif.

The overall trend suggests an increase in the edit distance of the motif as the sequence length increases from less than 100 up to 200, after which the edit distance decreases. The reason could be that as the sequence length increases, and since our method’s performance does not decrease with sequence length because of more random bases in the sequence, the over representation of the motif is picked up more easily. The trend of the mean PWM divergence is consistent with this hypothesis.

Overall our method showed consistency throughout the length increase and even found the motif in all cases.

7.1.1 A Note On Performance Measures

The PWM divergence and edit distance have been shown to complement each other in the case of performance. For example, a motif in the experiment with one mismatch got a 0.25 score for the PWM Divergence measure, whereas

some motifs with edit distance of zero got a score of 0.35 as their divergence value. The reason is that some individuals only get the consensus form of the target motif right and ignore the other forms (motifs with the best entropy score of zero), whereas some others get several degenerate forms and have higher entropy values for each column. Consequently it is suggested that the combination of the PWM divergence score and the edit distance score be used to compare motifs.

7.2 Tompa's Dataset

We chose the same 3 datasets in Tompa's dataset that were previously used by the MOGAMOD algorithm. This would enable us to compare our algorithm to the other multi-objective program. Data from other motif finder algorithms (e.g. the binding sites identified and the position of the detected site) are also available in Tompa's benchmark, allowing us to do further comparison. First, some combinations of objectives are used to show the effectiveness of each on the dataset. Then the best combination is used to compare the motifs found by our tool to the other methods.

7.2.1 Objective Effectiveness Study

In this section we test the effectiveness of each objective while keeping two objectives constant. Unfortunately, since the PWMs and the consensus sequence of the known motifs are not readily available in the dataset we are unable to calculate the PWM divergence and edit distance to the known site as a means of calculating the effectiveness of each method. Thus our performance measure would be how well the best motifs in the archive match the motifs reported by the benchmarking suite and other algorithms.

Firstly, the Entropy and Last State Visit objectives are set as the base combination. This choice is due to preliminary experiments. The base combination remains constant throughout all the runs.

For each run only one objective is added and compared to the base combination. Each run is carried out 10 times using the same GA parameters as Table 7.1 and motifs of length 10. These 10 runs each produce an archive of best Pareto non dominated individuals in the run. At the end of 10 runs the archive is analyzed for found motifs. The size of the archive is not limited, and is usually from 3 to 7 depending on the objective used.

Table 7.2: Objective Effectiveness Comparison

	Objective Combination		
	Base	Base + Likelihood	Base + Clustering
Motifs	ATTTTTTTTT	CCGTCGCTCG	CATCCAGTTT
	AAAAAAAAAA	ACGCCGCGGG	TTTTCTCCTC
	TTYTTCYTTW	CGCGGCGGGG	CATCGGGATT

The clustering objective consistently created a larger archive size than entropy since it actually adds three objectives: the intra cluster, inter cluster sums of errors and the distance between the two cluster means. Some found motifs of each experiment are shown in Table 7.2. It was not expected that any of these objectives would get good results on their own. However, in comparing some of these results to the known motifs in this dataset clustering showed superior predicted motifs than the other two objectives and also showed much more diversity in the archive’s individual set.

The reason this experiment was carried out was to show how differently each objective covers the search space. For example the Likelihood objective may look for individuals that have a lower chance of producing a motif sequence given the dataset’s base composition statistical model, while the clustering objective would never give those individuals good scores. The use of these objectives that each consider one aspect of the motif finding problem results in more diversity in the population and prevents premature convergence.

7.2.2 Comparison to other programs

In this section we experiment with the yeast04r, yeasr08r and hm03r datasets in Tompa’s benchmark to compare our methods to the programs AlignAce, MEME, Weeder and MOGAMOD. 1000 individuals were evolved for 100 generations. All other parameters remain the same. Motifs found by our program and the other programs are listed in Tables 7.3, 7.4 and 7.5.

As seen in these tables, our algorithm was as successful as other algorithms in finding the motifs in the dataset. An advantage of our algorithm compared to others is that in each run many possible motifs are reported to

Table 7.3: Found Motifs In Yeast04R

Program	Motifs Found
Our Program	CTGGCATCCA, CCGCGGATCG, CCTTTTCTGG
Weeder	TTTTCTGGCA
AlignACE	CGGGATTCCA
MEME	CGGGATTCCCC
MOGAMOD	CGAGCTTCCACTAA, CGGGATTCCTCTAT

Table 7.4: Found Motifs In Yeast08R

Program	Motifs Found
Our Program	ACACCCAGAC
Weeder	ACACCCAGAC
AlignACE	CACCCAGACAC, TGATTGCACTGA
MEME	CACCCAGACAC
MOGAMOD	ACACCCAGACATC

Table 7.5: Found Motifs In Human03R

Program	Motifs Found
Our Program	TGTCTAGCTA
Weeder	TGATCACTGG
AlignACE	TGTGGATAAAAAA
MEME	AGTGTAGATAAAAGAAAAAC
MOGAMOD	TATCATCCCTGCCTAGACACAA, TGACTCTGTCCCTAGTCT, TTTTTTCACCA

the user. For example in the Yeast04r dataset Weeder found only one set of degenerate motifs and other algorithms picked up on another motif, whereas our approach had both sets in the archive.

Chapter 8

Improved Results

In the previous chapter initial experimentations were discussed. In this chapter new approaches are sought to improve the previous results.

8.1 Motif Database

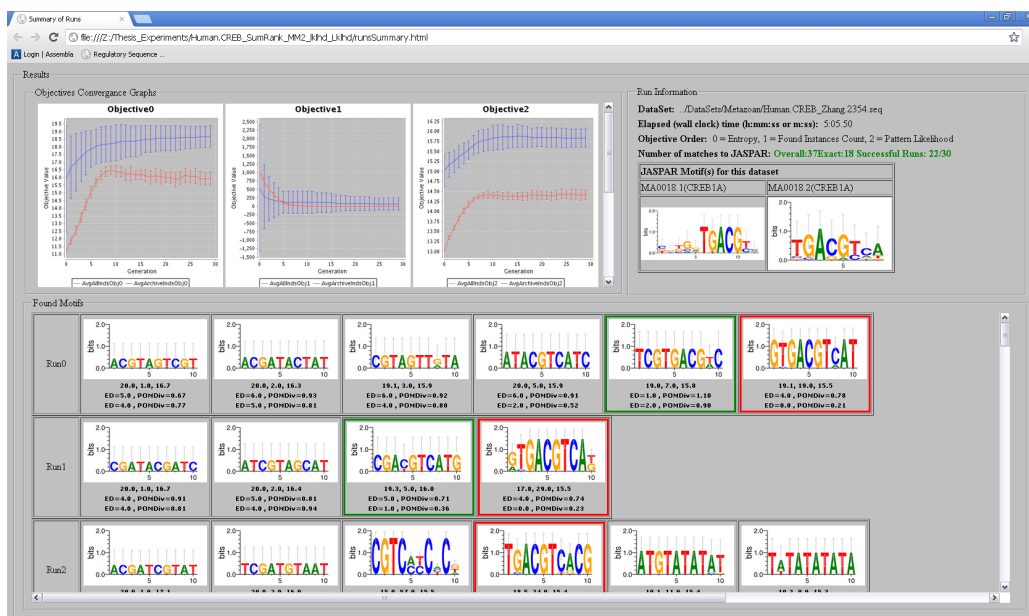
In previous experimentations the TRANSFAC database was used since the benchmarking suites used this database. Unfortunately this database's free version proved to be unusable for the purposes of this work since it was incomplete, slow and not available as a flat file. An alternative to this database which also holds motif information was sought and JASPAR [9] was chosen. This database as stated in its documentation "is a collection of transcription factor DNA-binding preferences, modeled as matrices". This is the only open source database in this scope which allows unrestricted access to the data it holds. Also, the ability of downloading a flat file of the entire database greatly simplifies the process of result validation.

8.2 Mass Experimentation Tool and Summary Page

A system was developed to enable mass experimentation of the possible parameter space and further approaches. The program would submit runs to two available clusters, one with 36 Intel corei7/PhenomII cores and the other with 36 Intel corei5 cores. After the submitted job returns the program ana-

lyzes the results by extracting run information as in average objective value per generation chart, average objective value of archive individuals chart, run time, number of successful runs, similarity to original motif in JASPAR, etc. After the data collection phase a summary page is generated which would show the results of the run in an HTML format. A snapshot of the system can be seen in Figure 8.1

Figure 8.1: Snapshot of Program Output



This system adds the following capabilities and parameter options.

8.2.1 Motif Representation

One of the shortcomings in the previous study was the visualization of the motif. The comparison between different approaches and methods proved difficult since the motif representation was its frequency matrix and consensus. A better representation called motif logo as explained in section 3.1, is then taken as the motif representation for the program output. This decision helps the comparison process between different runs of the algorithms plus the comparison between the algorithm and others which also take the same approach. For implementation a library created by [13] is used.

8.2.2 Ranking System

With the multi-objective approach, at the end of a run of the algorithm a family of results is shown to the user. Although in a multi-objective problem no trivial sort exists and if one did exist there wouldn't be a need for a multi-objective approach, some ranking methods are analyzed.

8.2.3 Background Model Additions

Different probabilistic sequence models were discussed in section 2.2.2. In order to test different models for their effect on the performance the new system allows IID, 1st, 2nd and 3rd-order Markov models for the background sequences.

8.2.4 Multi-objective Scoring Schemes

The multi-objective scheme used in chapter 7 was the normalized sum ranks. The capability of choosing other multi-objective scoring schemes was added to analyze the different multi-objective scoring schemes behavior. These schemes were discussed in section 2.1.2.

8.2.5 Convergence Charts

The program averages the entire generation's objective value for each objective per generation and then averages this value over all runs (number of executions of the algorithm per experiment). The depiction of the trend of progress during evolution shows how well the population converges and how the objective values relate to one another. Due to space issues these charts may have illegible labels for the axes. The axis labels would then remain constant and consistent during the runs with the X axis representing the "Objective Value" and the Y axis representing the "Number of Generations". This data series is drawn in blue.

There are also bars representing the average standard deviation of the individuals per generation throughout the run. While this statistic does not show the statistical standard deviation for a number of sets based on population-based statistics, it is used to depict the variation of the objective value in each generation throughout the runs.

In addition, the charting section of the program graphs another data series

in red which depending on the multi-objective scoring system used can be one of the following:

- Summed Ranks: The custom archive set.
- SPEA2: The archive set used by SPEA2 as discussed in section 2.1.2.
- NSGA-II: The Pareto front of the population in the last generation of the run.

In order to simplify the analysis, all the objectives in all the experiments conducted are going to be maximized. Even though objectives like Entropy and Likelihood should actually be minimized, they are converted into a value that could be maximized to minimize the actual objective. For example when working with the Entropy objective the value of zero is the best possible value. But by choosing the following formula for the objective we could have the same effect by maximizing it:

$$Entropy = (2 \times m) - Ind_E$$

Where m is the pattern length and Ind_E is the raw entropy value for individual Ind .

Therefore, it is very important to remember this setup during the experiments and know that *higher* values are better.

8.2.6 Matches to JASPAR

The program checks each motif found in the runs with the JASPAR database. The motifs are compared with two performance measures also used in previous experiments, the edit distance and the PWM divergence. If the PWM divergence of a motif is less than 0.30 from any of the JASPAR motifs or if the edit distance is less than $\lceil \frac{M}{4} \rceil$, where M is the length of the JASPAR's motif, then the motif is considered a match. In addition, if the PWM divergence from the JASPAR motif is less than 0.24 then the found motif is considered an exact match. In the case that JASPAR's reported motif is of differing length to the motif found, a sliding window of the width of the smaller of the two is moved on the motif with higher length. For each pattern enclosed in the window in the longer motif the values for PWM divergence and edit distance are stored and the lowest of these values is reported.

In order to differentiate found matches to JASPAR, normal matches are

marked with an outlining green square around the motif and an exact match is marked with a red square around it.

8.3 New Experiments

8.3.1 IID vs Markov Models

The likelihood objective looks for the probability of occurrences of a pattern. This objective's mechanism was discussed in 2.2.1.

In the IID model the assumption is that each of the positions in the motif is statistically independent of other positions, for example the occurrence of C after A in the pattern ACGACC does not have any significance. Apparently this assumption is a very restrictive one to the motif model. Most motif finders that base their algorithm on a probabilistic model may use an IID approach since the change to a Markov model over complicates the parameter inference machine learning approach which is already quite complex. In this algorithm a move to a higher order Markov model does not require much change and can be added to the algorithm by first analyzing the background sequence data to extract a probabilistic model and a mechanism to score observations.

Likelihood Calculation Method

In the previous section the IID model was extracted from the background by counting the number of occurrences of the bases and dividing them by the overall length of the input sequences. In the case of a Markov model depending on the order, the number of occurrence of a base followed by 1,2 or 3 or more bases has to be calculated. As an example for extracting the probabilities as a 2nd-order Markov model the process is as follows.

Firstly the IID and 1st-order Markov models are extracted from the input sequences since these models are needed to calculate the observation probability as stated in section 2.2.1.

Secondly in the case of the 2nd-order Markov model all possible overlapping 3-mers are extracted from the sequences keeping count of the number of occurrences that start with the first two characters. For example if the found 3-mers are ACC, ACG, ACC in different positions in the input sequence set then the probability of C after AC is $\frac{2}{3}$ and probability of G after AC is $\frac{1}{3}$.

If a pattern is not found in the input sequences then the probability of it is calculated using Laplace smoothing as described in section 2.2.3.

This section compares IID, 1st-order, 2nd-order and 3rd-order Markov models plus a case of Non-homogeneous Markov model. The more effective the background modeling, the more information is provided to the evolution by the likelihood objective.

IID

We start with results from the IID background model. The parameters used for this experiment are shown in Table 8.1. The graphs in Figure 8.2 have the results. The first thing to note in the graphs is that in a multi-objective

Table 8.1: Background Model Experiment Parameters

Parameter	Value
Number of Runs	30
Number of Generations	30
Mutation Rate	10%
Crossover	90%
Population	500
Selection Method	Tournament Selection (Size 3)
Multi-objective Scoring Scheme	Sum Ranks
Objectives	Found Motif Count + Entropy + Likelihood
Datasets	CREB_Zhang, HSF1_Page and GATA_Pauli
Motif Length	10

setting, the improvement of all objectives simultaneously is not always possible. As seen here, one objective improves at the expense of the other. Predicting which objective would dominate is not always straight forward. For example, in the case that one objective has a wider range of possible values and the fitness function is the multiplication of the objective values, one can predict that the objective with the higher range of values may be able

to dominate the other objective by dominating the fitness score. Conversely, in this experiment with the choice of normalized sum ranks (section 2.1.2) as the multi-objective scoring scheme, this prediction is not possible.

In addition, it is important to understand that the early degradation of the Found Motif Count objective as can be seen from other experiments throughout the rest of this chapter is actually desirable. The reason for this is that the initial individuals in the first generation are mostly highly variable NFS-SEMs as they have very high entropy and represent an undesirable highly degenerate motif. This happens although in the initialization stage the probability of a transition to the next stage or the first stage are both 0.5.

Furthermore, since the number of occurrences of the motif in the sequences was neither accepted as an input parameter to the algorithm nor was it enforced to be a percentage of the number of sequences, it is the multi-objective scoring scheme's job to produce a diverse set of individuals covering the entire fitness landscape in the Pareto front.

On another note, a quick comparison of the three different datasets' convergence rate truly shows the difficulty of this problem for the GA. Each of the three datasets behave differently in their evolution. In CREB dataset specifically as will be apparent on later sections, the improvement of the Entropy objective always dominates the Found Motif Count objective. This trend is broken in the IID background model experiment as shown in Figure 8.2(a). Strangely, this is the only dataset in the three that shows this effect. Some of the found motifs from the 30 runs are the motifs shown in Figure 8.3.

Notice in Figure 8.3 that all the motifs shown have mostly or only As and Ts in their logos for the CREB_Zhang dataset and for the HSF1_Page dataset. This is true for all the other motifs found in the datasets. Also, in GATA_Pauli, motifs mostly consist of Gs and Cs. This we would expect if the probability of occurrence of the bases with most occurrence in the found motifs is low in the input sequences. An analysis of the sequences' base composition reveals that this indeed occurs, with A and T having the lowest frequency in CREB_Zhang ($A=0.23$, $C=0.27$, $G=0.26$, $T=0.22$) and HSF1_Page (0.23 , 0.27 , 0.27 , 0.22), and Gs and Cs being the lowest in Elegans ($A=0.31$, $C=0.18$, $G=0.15$, $T=0.34$). This shows that the GA is successfully converging to what seemed the best with the information we supplied with the IID model, but the fact that none of the reported motifs are the known motifs reported in JASPAR means that the IID model is insufficient to model the known motif.

In order to solve this issue, a Markov model to provide more statistically

Figure 8.2: IID background model convergence charts. The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood

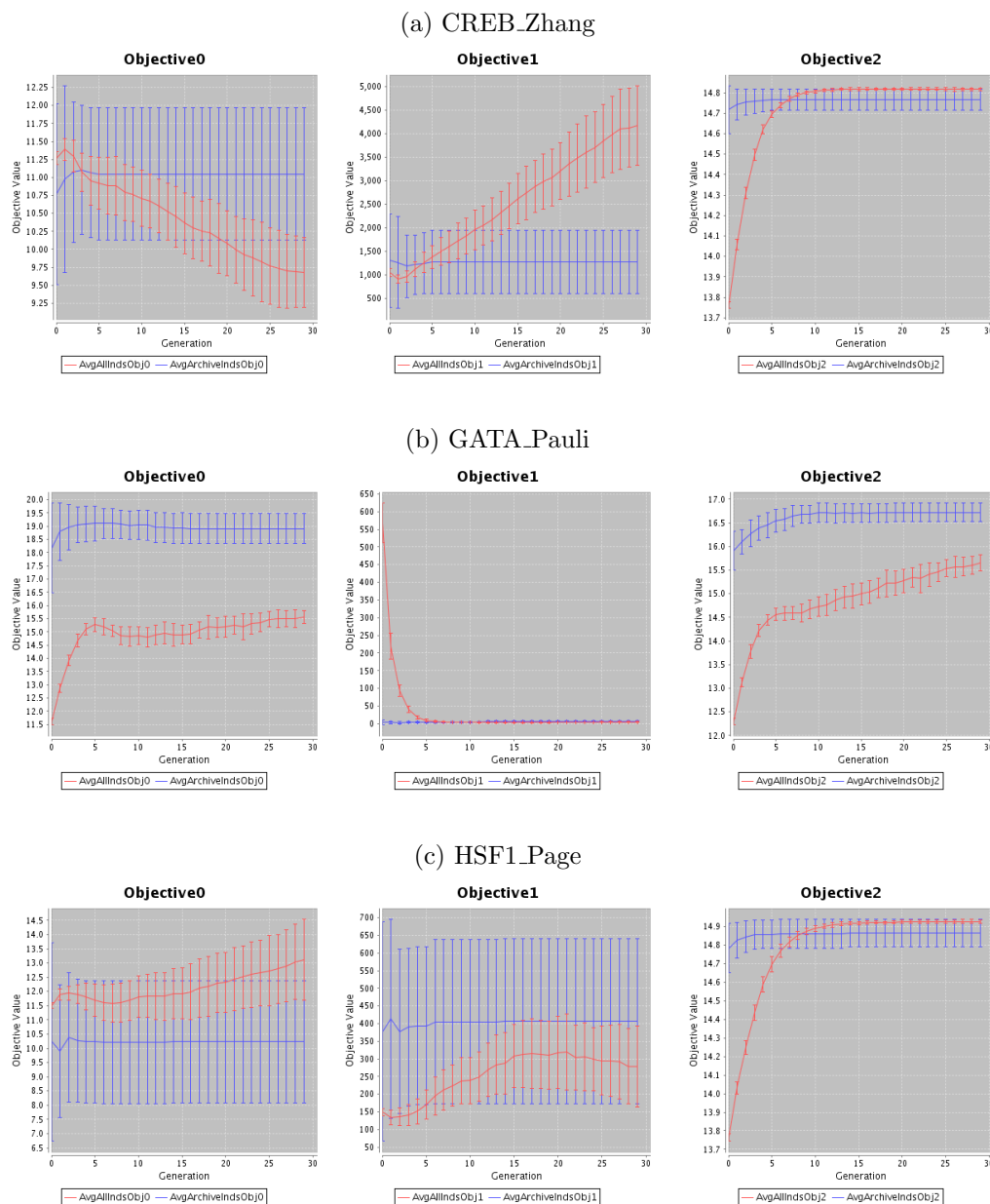
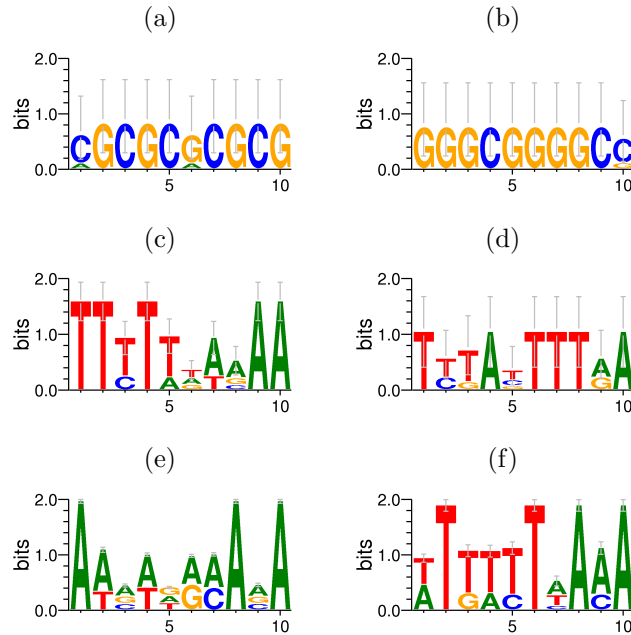


Figure 8.3: Found motifs in the human CREB TF dataset using an IID background model. (a) and (b) are from the HSF1_Page dataset, (c) and (d) are from GATA_Pauli and (e) and (f) are from CREB_Zhang



accurate information to the GA.

1st-order Markov

In this section the 1st-order Markov model is chosen for the probabilistic model. Also, the non-homogeneous Markov model that was discussed in 2.2.2 is tested.

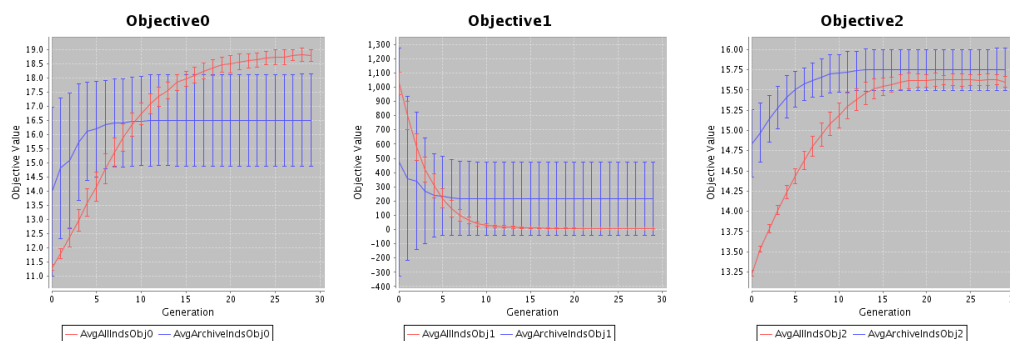
Figure 8.4 has the results for the 1st-order Markov model. These convergence charts still show the different evolution trends in different datasets although the objective domination trend is consistent throughout the database. The Entropy objective improves at the expense of Found Motif Count.

The likelihood objective is still unable to guide the search towards the known JASPAR motif, except in one case throughout the runs in the CREB_Zhang dataset. This motif in addition to some other examples of found motifs are shown in Figure 8.5.

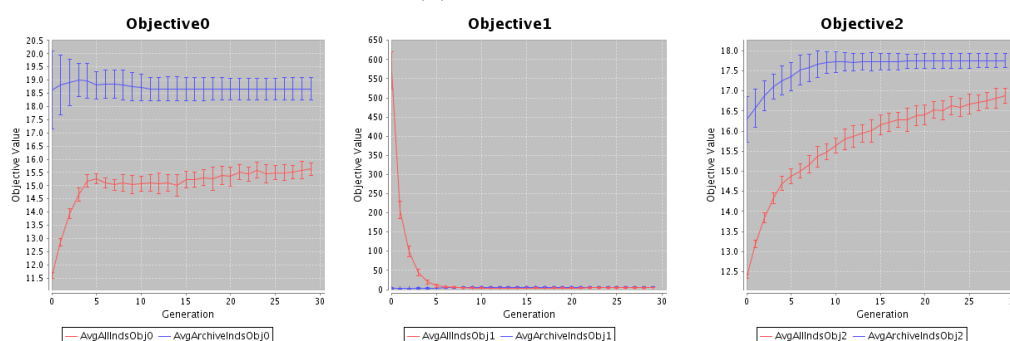
The motifs in Figure 8.5 show a more complex pattern rather than repe-

Figure 8.4: 1st-order Markov background model convergence charts. The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood

(a) CREB_Zhang



(b) GATA_Pauli



(c) HSF1_Page

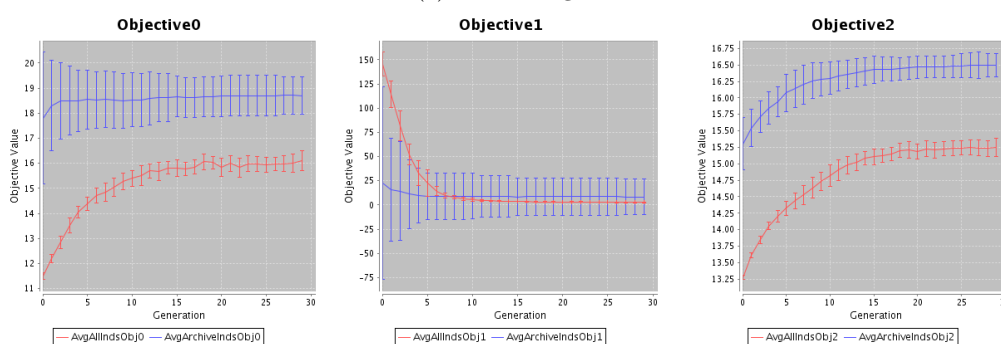
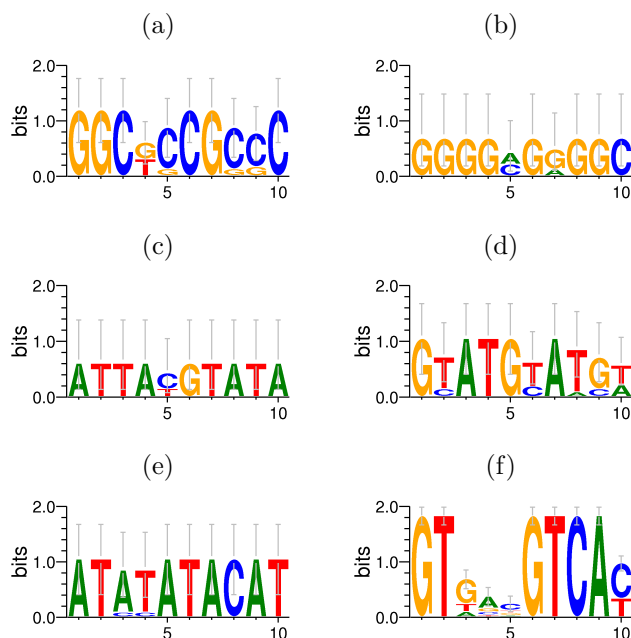


Figure 8.5: Found motifs in the human CREB TF dataset using an MM1 background model. (a) and (b) are from the CREB_Zhang dataset, (c) and (d) are from HSF1_Page and (e) and (f) are from GATA_Pauli

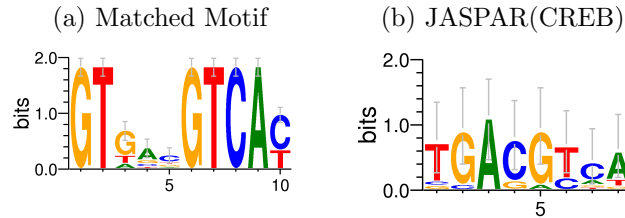


titions of the bases with least occurrence rate. Although more complex these motifs are now repetition of a pattern of length two with least occurrence (e.g. TATATATA). Although there are motifs with this recurring patterns as in the TATA-box motif which has a recurring “TA” pattern but most motifs consist of more complex mixture of bases.

Even though the algorithm is still unable to pick up on complex patterns, the motif shown in Figure 8.5 section (b) is actually a match to the JASPAR database’s motif but it was only one match throughout the entire 30 runs and it is not an exact match. This motif in a side by side comparison is shown in Figure 8.6.

The found motif has found the consensus sequence for the motif TGACGTCA. But this is not an exact match since the frequencies of the bases are not correctly identified. This is not surprising since at no point in the algorithm are these frequencies optimized for the best fit. These frequencies are merely a maximum likelihood estimate extracted from the input dataset.

Figure 8.6: Comparison between JASPAR’s reported motif for CREB_Zhang and found motif



Next we test the non-homogeneous Markov model. Figure 8.7 shows the convergence charts for this experiment. A comparison of these graphs and the motifs found in the run shows that there is no difference either in the evolution of the population or the motifs found in the archive.

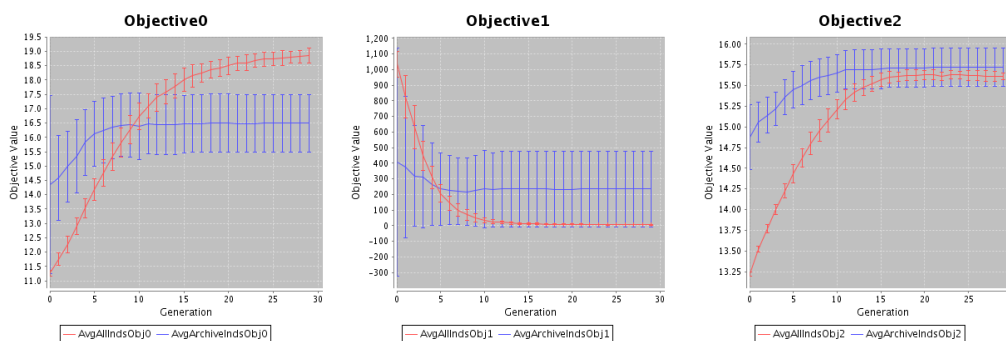
One reason for this could be that since all overlapping subsequences of a given length are analyzed for the modeling process, the position of a certain base (in case of an IID model) or base pairs (in case of the non-homogeneous Markov model) does not add any more information. For example if we have a pattern “ACGCACTTGA” and we try to model 3-mers, the third base “G” would be once regarded at position 0 (GCA), once at position 1 (CGC) and once at position 2 (ACG).

This process is also carried out for all other bases and since each base appears once in each position, it renders the position specific modeling useless. The ability for each base to appear in each position is true with the exception of a few bases positioned at the very start and the very end of the sequence (in the example AC and GA). These small percentage of bases in the sequence, due to their position, could not appear at certain locations in the overlapping subsequences which are considered for probabilistic modeling. But due to their small numbers, their effect on the overall statistical model is negligible. This is especially true for datasets with long sequences as in the datasets studied here.

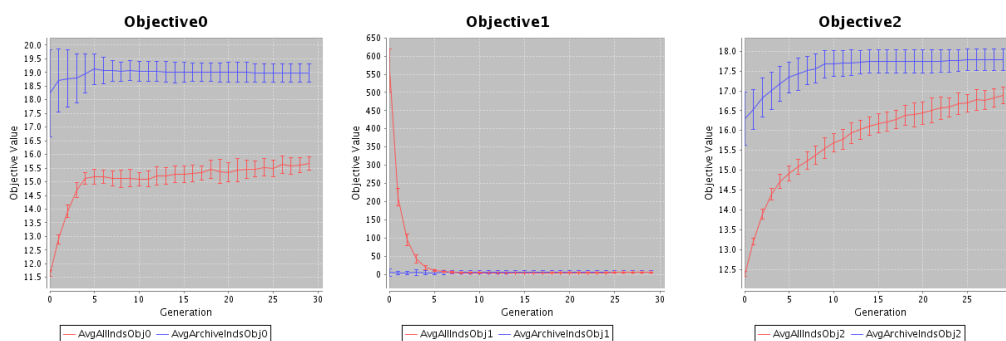
In short, since the non-homogeneous Markov model shows no improvement to the stationary-Markov model, this model is not further analyzed for other higher-order Markov models.

Figure 8.7: Non-homogeneous 1st-order Markov background model convergence charts. The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood

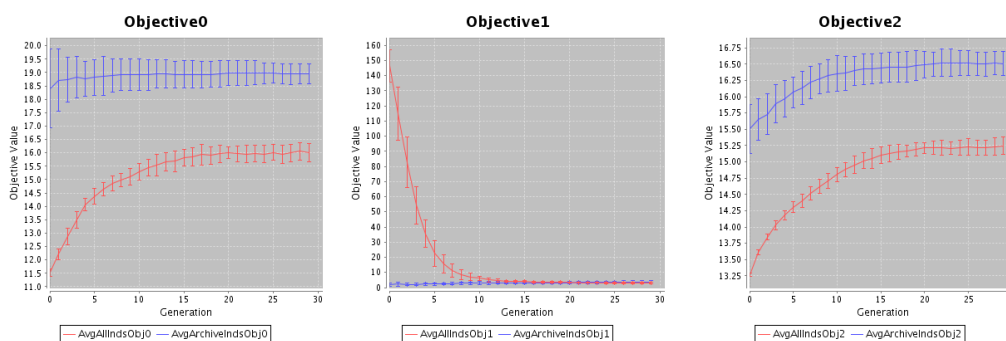
(a) CREB_Zhang



(b) GATA_Pauli



(c) HSF1_Page



2nd-order Markov Model

In this section the algorithm is run with a 2nd-order Markov model for the probabilistic background model. The convergence charts are reported in Figure 8.8.

The convergence charts in this section follow on the previous trends of the Entropy objective and the Likelihood improving in the cost of the Found Motif Count. In previous experiments (8.4, 8.7), the Entropy objective in the archive was even surpassed by average individuals. This means that the individuals in the archive were more degenerate motifs than average individuals in the population. But in the case of the 2nd-order Markov model in Figure 8.8 the evolution was able to preserve the archive over the average individuals and also managed to keep the found motif count higher too which shows an overall superior evolution.

Figure 8.9 shows the two motifs found in the run accompanied by JASPAR's reported motif for the dataset. The first motif in this figure is actually a match with an edit distance of one, since starting from position 0, AGGTAC was recognized which is different from the solution (AGATAG) in only one base. The other motif is another prevalent motif in the runs. Although the size of the motif to be found is an input parameter, this parameter was not changed to match the solution motif in JASPAR since the algorithm is designed to find unknown motifs of unknown length. Also the solution was not used in any way during the evolution process and it is only used for performance feedback.

This JASPAR motif was found 3 times out of the 30 runs.

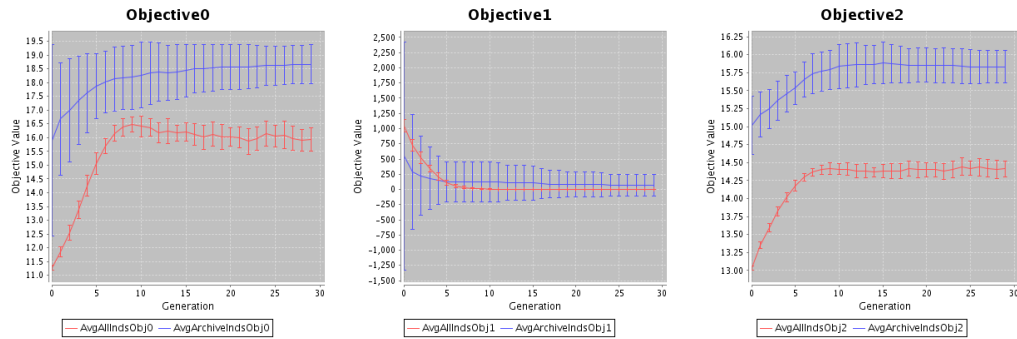
Figure 8.10 has the result for the CREB_Zhang dataset. This is by far the most successful run and a substantial improvement over the previous experiments. In this dataset the solution was found in 22/30 runs. Also out of the 37 overall matches, 18 are exact with PWM divergence of less than 0.24.

The success of this run shows that the 2nd-order Markov model was indeed able to effectively pick out the statistically overrepresented pattern.

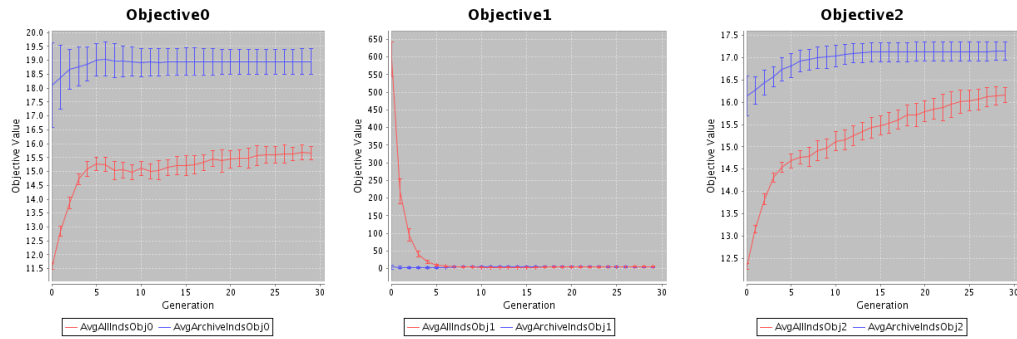
Figure 8.11 shows that the HSF1 dataset is a hard dataset for this problem. Although the motif was found in the other datasets with different success rates, this motif was not found in this dataset. On the other hand the reason for this could be realized by looking at the second motif in this figure. This motif was the same motif found in CREB_Zhang. This means that this is an actual motif that the algorithm detected to be more statistically over

Figure 8.8: 2nd-order Markov background model convergence charts. The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood

(a) CREB_Zhang



(b) GATA_Pauli



(c) HSF1_Page

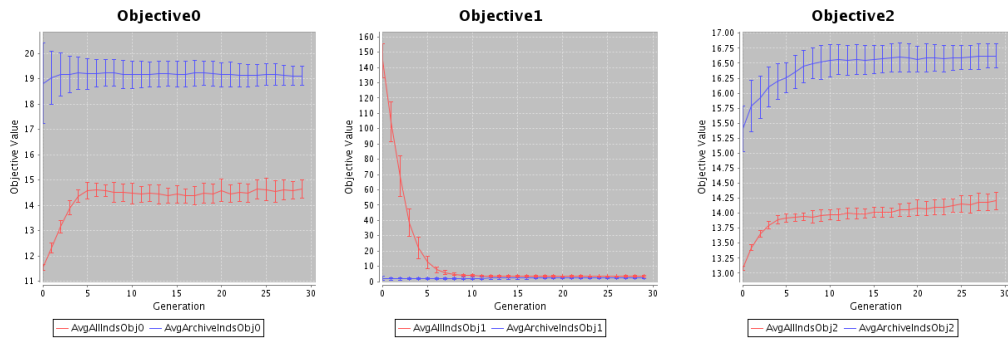


Figure 8.9: Found motifs in the GATA_Pauli TF dataset using an MM2 background model. (a) and (b) are two found motifs and (c) is the JASPAR's reported motif for the dataset.

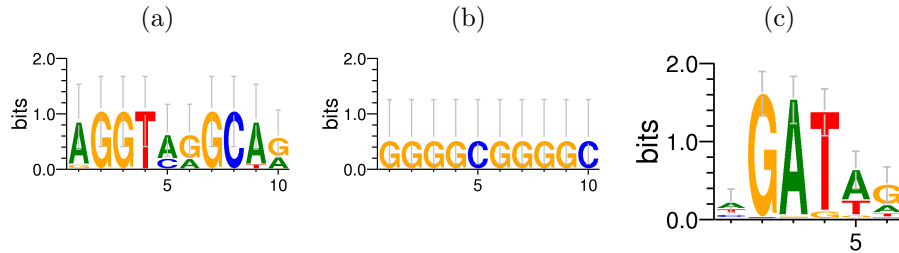
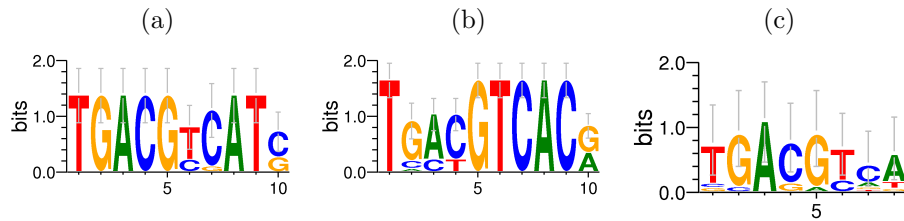


Figure 8.10: Found motifs in the CREB_Zhang TF dataset using an MM2 background model. (a) and (b) are two found motifs and (c) is the JASPAR's reported motif for the dataset.



represented in the dataset.

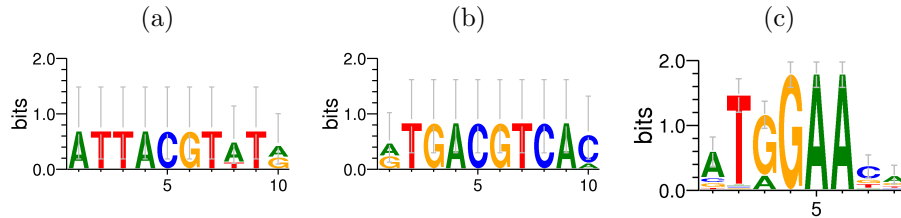
3rd-order Markov Model

In this section the algorithm is run with a 3rd-order Markov model for the probabilistic background model. The convergence charts are reported in Figure 8.12.

This shows an evolution trend of high similarity to the previous section. Although one might expect to get even better results in the found motifs, this is not the case.

Figure 8.13 has 2 examples of the motifs found in each dataset for this experiment. The algorithm failed to find JASPAR's motif for data sets HSF1_Page and GATA_Pauli. The motif for CREB_Zhang's dataset, which was successfully found in the previous section, was also found in this experiment but with limited success. There are 2 overall matches none of which are exact.

Figure 8.11: Found motifs in the HSF1_Page TF dataset using an MM2 background model. (a) and (b) are two found motifs and (c) is the JASPAR’s reported motif for the dataset.



The matches were found in two separate runs.

The reason for the deterioration in performance could be the length of the motif. The length of the motif to look for in the experiment is set to 10. It may be the case that a 3rd-order Markov model, which would calculate the probability of overlapping 4-mers in the motif, is too high for the motif with length 10 and modeling the sequences with that order would actually cause loss of information rather than information gain.

Discussion

In this section different probabilistic models for the background sequences and the motif were analyzed for the motifs they find. The improvements to the found motifs were shown when moving from the IID model toward higher order Markov models stopping at the 3rd-order Markov model. Consequently, for more extensive runs over the benchmarking suite the 2nd-order Markov model is going to be applied for the motif and the background sequences’ probabilistic model.

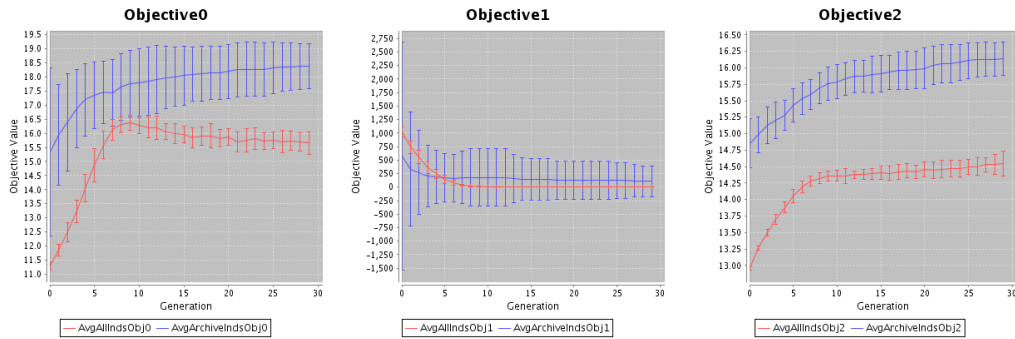
8.3.2 Multi-objective Scoring Scheme Comparison

The system is capable of running the algorithm using Summed Ranks with custom archive, SPEA2 and NSGA-II as the scoring scheme for fitness evaluation. In this section these different methods are compared on different datasets and the results are analyzed.

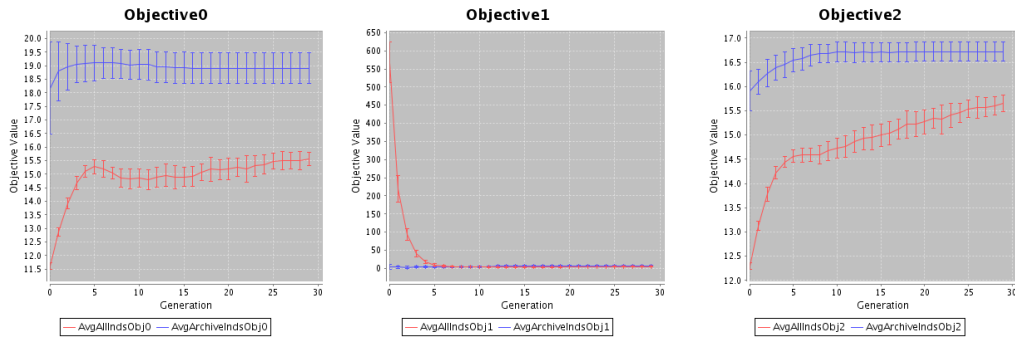
Although all these methods report the Pareto front as their final best results, SPEA2 and NSGA-II have a diversity strategy to spread the final answer

Figure 8.12: 3rd-order Markov background model convergence charts. The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood

(a) CREB_Zhang



(b) GATA_Pauli



(c) HSF1_Page

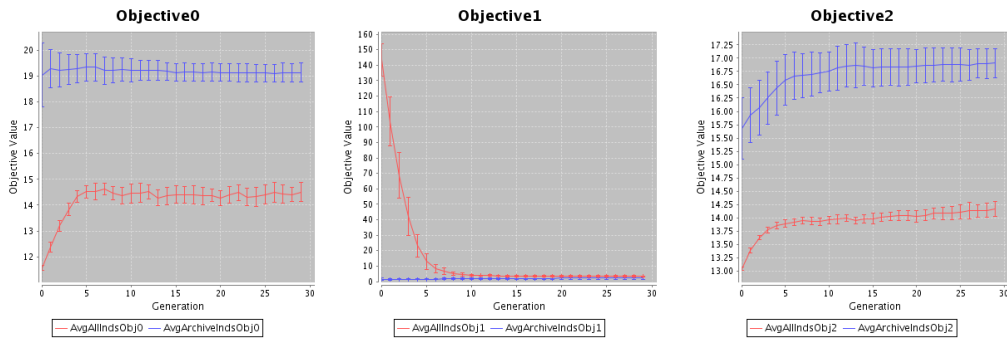
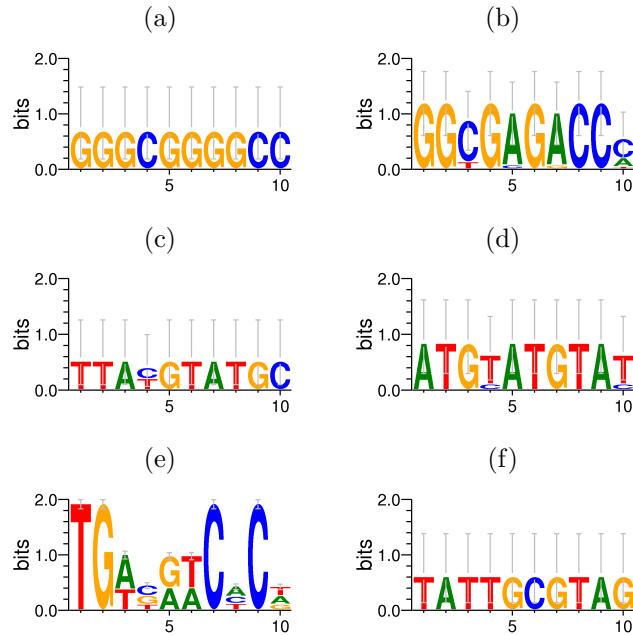


Figure 8.13: Found motifs in the human CREB TF dataset using a MM3 background model. (a) and (b) are from the CREB_Zhang dataset, (c) and (d) are from HSF1_Page and (e) and (f) are from GATA_Pauli



throughout the Pareto front and prevent niches. The parameters used for this experiment are shown in Table 8.2.

The comparison looks for the diversity of the motifs found plus the evolutions trends and the solutions found. Firstly their convergence trends are analyzed. Convergence charts for summed rank have already been included in Figure 8.8. The convergence charts for SPEA2 and NSGA-II are shown in Figures 8.14 and 8.15.

The graph in Figure 8.14 is almost incomprehensible due to the high similarity between the archive set and population values and the large variation in them. The overlap is consistent with the scoring mechanism which populates the next generation directly and only from archive individuals. This results in the population and archive objective values always being close.

The diversity strategy also keeps adding to the variation in individuals as the generations grow. The second objective (Found Motif Count) also shows improvement while the two other objectives fluctuate.

Table 8.2: Relative Entropy vs Entropy Experiment Parameters

Parameter	Value
Number of Runs	30
Number of Generations	30
Mutation Rate	10%
Crossover	90%
Population	500
Selection Method	Tournament Selection (Size 3)
Background Model	2nd-Markov Model
Objectives	Found Motif Count + Entropy + Likelihood
Datasets	CREB_Zhang, HSF1_Page and GATA_Pauli
Motif Length	10

The evolution does not look promising. No objectives is converging or even showing a trend that suggests convergence, except the second objective which as mentioned earlier is expected to decrease in value in the few starting generations and then converge. Overall the two fluctuating objectives do not improve at all and it is expected from this graph to see solutions that are highly degenerate motifs that are very high in entropy and also rather high in likelihood.

NSGA-II's convergence charts in Figure 8.15 show only one graph per objective. The reason for this is that NSGA-II does not have an archive and the entire population is evolved with only a diversity strategy in mind. In the case of diversity this approach does not produce as diverse individuals, but the diversity does not deteriorate with evolution either.

Next, motifs found by any of these algorithms are reported. Since these motifs may not be the same families in each run, 3 motifs from 3 runs are chosen and reported in Appendix A.

Figure A.1 shows the motifs found in CREB_Zhang with the NSGA-II approach. This approach finds an overall 51 matches to the JASPAR database in 30 runs two of which are exact matches. This means that 20 runs out of 30

Figure 8.14: SPEA2 convergence charts. The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood

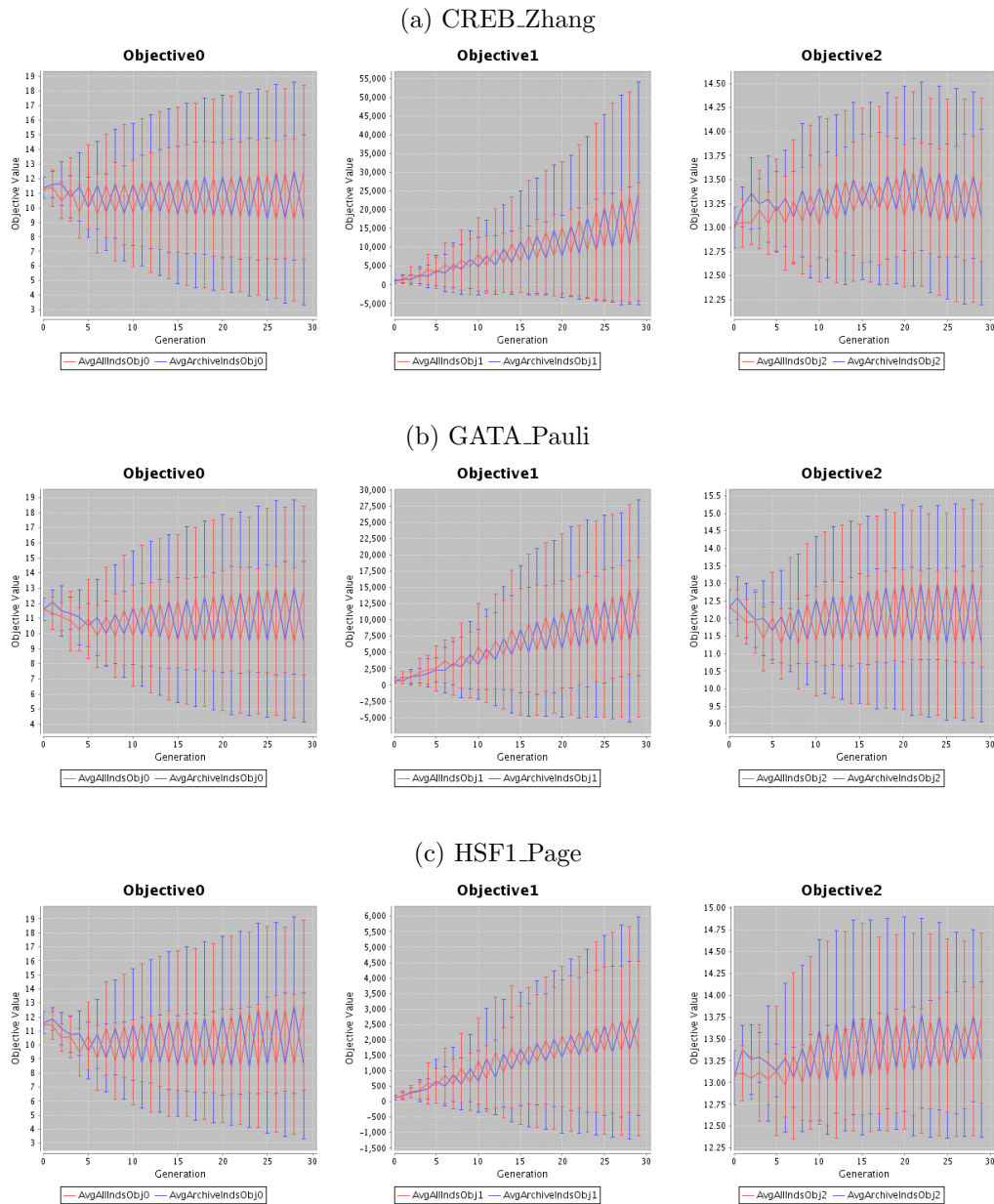
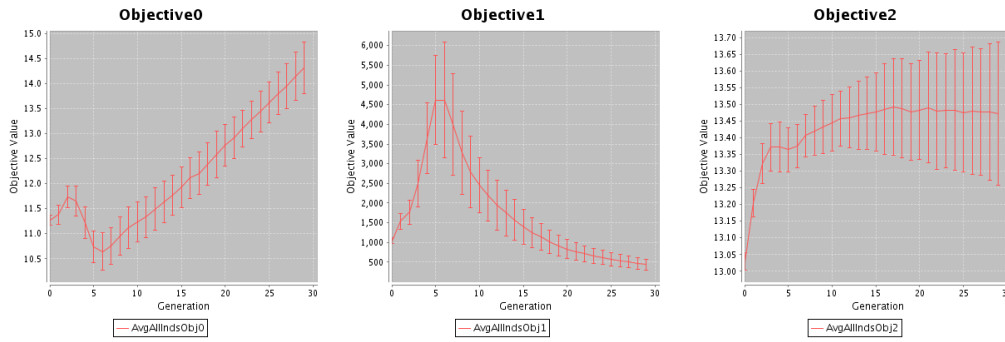
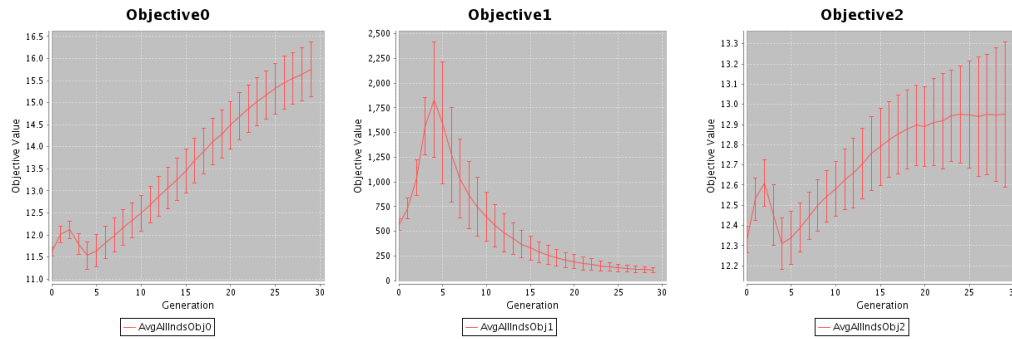


Figure 8.15: NSGA-II convergence charts. The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood

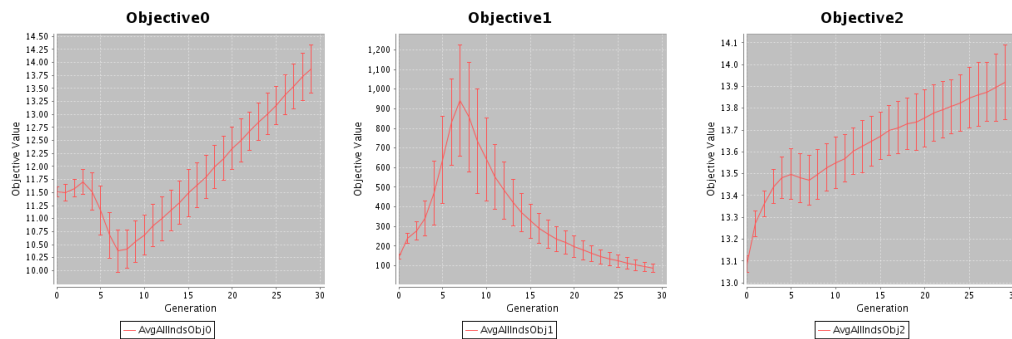
(a) CREB_Zhang



(b) GATA_Pauli



(c) HSF1_Page



runs were successful in finding a match for the motif. This approach shows very good diversity in the final results. The diversity strategy effectively spreads all the individuals throughout the Pareto front so that the objective values show good variance for the final solutions.

What makes this algorithm perform poorly in finding exact matches of the motif is the diverse nature of the results with different values for the Entropy objective. Usually the motifs found in JASPAR do not have very high entropies, i.e. are not extremely degenerate, and usually create 10 to 15 different possible forms and not hundreds of different forms. This characteristic of the motif prevents a certain region in the Pareto front (those with high entropy values) to gain the exact match score.

Since the shortcoming mentioned above is the specific characteristic of the search space, this shall not be thought of as a drawback to the algorithm since the algorithm's task is to produce a good spread of the Pareto front which it successfully does in the case of the CREB.Zhang dataset.

Figure A.2 has the found motifs in the GATA.Pauli dataset. There were 14 successful runs out of 30 and 19 overall matches, none of which are exact. The solutions still show very good diversity both in the motifs found and in the objective value's spread. In NSGA-II the mechanism for diversity tries to keep the distances of individuals from one another as high as possible. Although there are families of solutions in each run with the same genome but different objective values, the algorithm still shows a good diversity in the genome of the motifs found.

The results for the run on the HSF1.Page dataset are shown in Figure A.3. JASPAR's motif was not found in this dataset but some found patterns are reported in this figure.

SPEA2's performance on different datasets was poor as expected. Most motifs found in each run are mostly the same motif (same genome) but with different objective values. This algorithm was unable to find the solution in HSF1.Page and CREB.Zhang but some other found motifs are reported in Figures A.4 and A.6. An inspection of the motifs found in different runs shows that the algorithm is truly prematurely converging to a local maxima since in each run different sets of motifs are reported and not one single family of a motif.

It is surprising that this algorithm was able to find the motif in GATA.Pauli with 273 overall matches and 27 successful runs out of 30, but a closer look reveals that the reason for this higher overall number of matches is that when a match occurs the individual is usually present throughout the archive at the

end of the run. This means that if the archive is of size 10, on average, 9 other individuals are simply the same motif repeated. This shows that although the algorithm has a very large variance in the objective values it has little to no variance in the genome diversity of the members of the archive. Summed ranks showed reasonable convergence in its convergence charts. This algorithm outperforms the other two in the CREB_Zhang dataset with 39 overall matches, with 14 exact matches and 23 successful runs. Also the motifs found in each run are roughly of the same set which shows that there is no premature convergence to local maxima. NSGA-II also shows good results in this dataset with 51 matches and 2 exact matches in 20 successful runs. SPEA2 however couldn't find the JASPAR motif in this dataset.

Discussion

Overall NSGA-II and Summed Ranks showed promising performance in finding a good spread of the Pareto front with NSGA-II showing a better spread of objective values and Summed Ranks showing a better spread of the motif diversity (due to the custom archive). SPEA2 does not show good evolution trends but was able to find JASPAR reported motif in the dataset in one of the datasets tested.

8.4 Comparison to Other Motif Finders

The algorithm was tested in its early stages to some other algorithms reported in [50] using the Tompa dataset described in section 7.2.2. This dataset contains very short sequences for the input sequences. With current sequencing technologies both the length and the number of sequences available to biologists are growing and as a result benchmarking suites with longer sequences are needed to test modern motif discovery algorithms. The Metazoan compendium discussed in section 6.3 is an example of these newer benchmarking suites.

The datasets used in this section are chosen from this compendium with the criteria that the motif looked for is available in JASPAR. This is because we need to compare the found motifs with a solution motif and since we do not have access to the TRANSFAC database, only datasets containing motifs with their information available in JASPAR can be used. This criteria leaves us with 9 datasets. In this section our algorithm is compared to some

Table 8.3: Comparison to Other Algorithms Experiment Parameters

Parameter	Value
Number of Runs	30
Number of Generations	60
Mutation Rate	10%
Crossover	90%
Population	500
Selection Method	Tournament Selection (Size 3)
Background Model	2nd-Markov Model
Objectives	Found Motif Count + Entropy + Likelihood
Motif Length	10

other algorithms on these datasets. The results for the other algorithms are extracted from the papers discussing the benchmarking suite [30].

The parameters used for this section’s experiments are shown in Table 8.3. The background model is a 2nd-order Markov model as this model was the most successful probabilistic model in experiments conducted in section 8.3.1. The multi-objective scoring scheme was shown to be a harder choice in section 8.3.2. The NSGA-II algorithm found a better spread of the data while the summed ranks found more accurate answers. Also the objectives discussed in 7.2.1 showed that both likelihood and likelihood + clustering are effective with different reported motifs.

For this to be a fair comparison, these two algorithm parameters are fixed to the values with the best scores so far. Since NSGA found a better spread of the data this scheme is chosen over Summed ranks. Also, the likelihood objective is chosen as the best overall. Nevertheless, the runs are also done with the likelihood + clustering objective, and the results are reported to further analyze the algorithm’s behavior using this certain objective combination.

The results are shown in Tables 8.4, 8.5 and 8.6. The table cells marked with “*” signify successful matches to the JASPAR motif. The matching criteria is as discussed in section 8.2.6. The convergence charts can be found in Appendix B. With a look at these charts it may be deemed that the GA has

not converged yet and it may be better to continue the evolution for longer, but some experimentation revealed that if the evolution continues, the only individuals left in the Pareto front would be one or two outliers with very high values for specific objectives but not desirable overall.

In HNF1a_Odom, Amadeus and Trawler are the two algorithms to find the biological motif embedded into the dataset. Our algorithm was unable to find the motif but the best match is shown in Table 8.4.

In E2F_Ren, our algorithm found 13 overall matches. This motif has many entries in both JASPAR and TRANSFAC. The overall matches are the result of comparisons to 5 different JASPAR motifs. The algorithm found these matches in 6 different runs out of 30. From the other algorithms Amadeus, Weeder and Trawler were also successful.

In the HSF1_Page, the algorithm was unable to find the motif. The closest motif found is reported. Interestingly, this is the case where the clustering + likelihood was able to find the motif, although only once in the 30 runs. This shows how hard it is to choose one combination of the parameters as the best, since each may search the fitness landscape using different strategies that may work in certain datasets and not in others. CREB_Zhang was the most successful dataset for our algorithm. As also reported in section 8.3.2 the algorithm was able to obtain exact matches to the JASPAR motif. Other algorithms successful in this dataset are Amadeus and Weeder. AlignAce and MEME were not able to finish in a reasonable time [30].

The GATA_Pauli dataset's JASPAR motif was matched by our algorithm in 13 runs out of 30 with 23 overall matches. YMF and Amadeus are the other algorithms successful at finding the motif.

The Myc_Oryan's reported JASPAR motifs were not located in the dataset. The only program to find the motif is Weeder. Interestingly in this dataset the likelihood + clustering combination found the motif with 3 overall matches in 30 runs.

The ETS1_Hollenhorst also proves a difficult dataset for our algorithm. The algorithms able to find this motif are Weeder and YMF. The motif that is reported for our algorithm in the figure though is an interesting find. Since we already have the consensus for CREB we were able to notice it in the results quite clearly in each run. This shows that the CREB transcription factor also works with ETS1 in regulating the gene downstream of the promoter sequence being probed. This shows that this algorithm has great potential for the field of composite motif discovery in which the coordination between transcription factors in the regulation initiation complex is studied. This is

accounted for by the multi-objective design which would produce more than one motif in each run.

In the MEF2_Blais dataset, the motif was elusive to all the algorithms. But again our algorithm picked up the CREB pattern in the sequences, which is reported in the figure.

Our algorithm found the motif with 12 overall matches in 11 successful runs in the SRF_Cooper dataset. Amadeus, Weeder and Trawler are the other successful programs.

Table 8.4: Results of Different Algorithms on the Metazoan Compendium

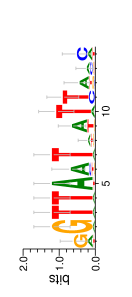
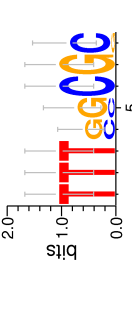
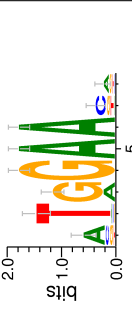
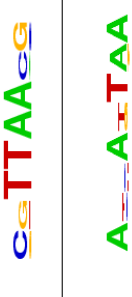
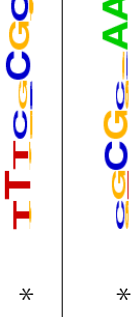
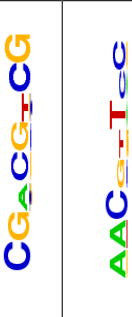
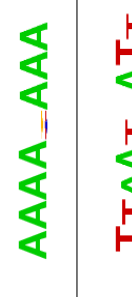
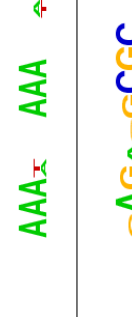




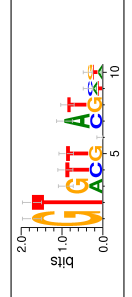
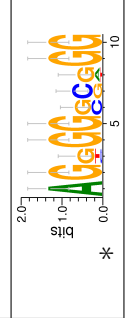
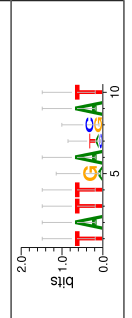






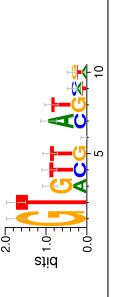
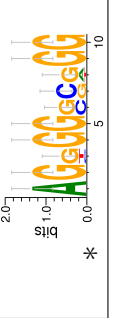
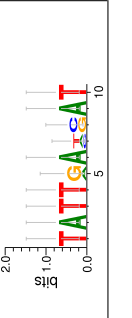
Dataset	HNF1a_Odom	E2F_Ren	HSF1_Page
JASPAR			
Weeder			
Amadeus			
AlignAce			
Trawler			
MEME			
YMF			
Our Algorithm			

Table 8.5: Comparison Results on the Metazoan Compendium

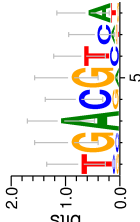
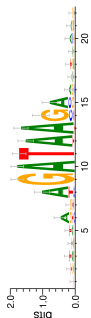
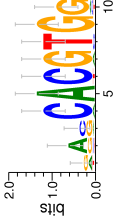












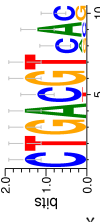
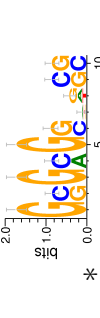
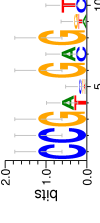
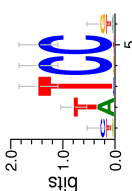
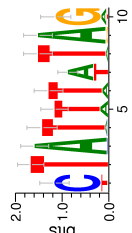
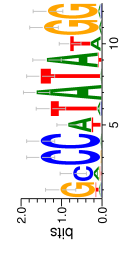
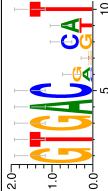
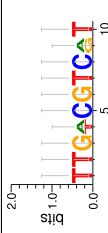
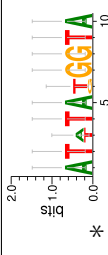
Dataset	CREB_Zhang	GATA_Pauli	Myc_Oryan
JASPAR			
Weeder	* 	* 	* 
Amadeus	* 	* 	
AlignAce	did not finish	did not finish	
Trawler	N/A	N/A	
MEME	did not finish		
YMF	did not finish		
Our Algorithm	* 	* 	

Table 8.6: Comparison Results on the Metazoan Compendium

Dataset	ETS1_Hollenhorst	MEF2_Blais	SRF_Cooper
JASPAR			
Weeder	CTTCC₂AG	GGG₁T₂GTG	CGTATACG
Amadeus	CG₂AA₁T	CTAA₂CC	CTTA₁AA
AlignAce	did not finish	GGG₂I₁GGG	G₁G₂ GGG G GG
Trawler	N/A	TGGGT₂CC	CTTA₂AGG
MEME	CGTA₂GT	CC₂GGCG	G₂ACGTAC
YMF	CG₂AA₁GG	CCCACCCC	CCC₂CCCC
Our Algorithm			

Chapter 9

Conclusion and Future Works

9.1 Summary

In this work, we implemented a de novo motif discovery tool using a GA to train NFS-SEMs with a multi-objective fitness scoring scheme. The experimentations started with SEMs to classify DNA sequences.

In the classification problem(section 4.2.5), the SEMs were tested on their ability to distinguish between sequences with different GC-content. This was shown to be analogous to the problem of discovering patterns containing motifs with more than 10 occurrences, since this level of repetition of the motif changes the GC-content considerably. The GA did not even need evolution and found the best individual mostly in the first few generations. This best individual could be evolved with only two states for the SEM. This two-state SEM(Figure 4.4) easily reached 100% accuracy.

The next problem removed the GC-content effect of the motif by only allowing it to either not appear, or to appear only once in a set of sequences. The experiments showed individuals with more than 5 states are able to reach 100% accuracy. Since the SEM is no longer picking up on the GC-content, it is actually finding the pattern. The work continued by introducing a new type of SEM which would make pattern extraction straight-forward. This SEM was called the NFS-SEM(section 4.3) since it was only allowed to make transitions to the next or the first state. The new type was able to obtain 95% accuracy and offer the extraction of the motif too.

In order to test the NFS-SEM to extract the actual pattern, firstly the approach was elaborated in Chapter 5. This approach, was used in a new set

of experiments in which the motif was embedded in sequences of variable length. This motif was inserted twice in sequences called the positive set and not inserted in negative sets. The two embedded motifs were degenerate forms of a chosen biological motif from the literature. The results(section 7.1) showed that the NFS-SEM is able to successfully pick out the pattern. The first biological benchmarking suite used was introduced in [50] and contained rather few and short sequences per dataset. This dataset was used to test different objective combinations for the motifs they find. This benchmarking was mostly used to analyze how each objective combination searches the space.

After this, a system was developed for mass experimentation on a cluster. The new experiments regarded different entropy approaches, background models and multi-objective scoring schemes(section 8.3). Finally the best combination of parameters was used on the Metazoan compendium and the results were reported(Table 8.4, 8.5, 8.6).

While Amadeus is tied with Weeder on the set of datasets analyzed here, it was the strongest motif finder reported in the entire Metazoan compendium. Even so, Amadeus uses a very powerful method and is a state of the art motif finder. It is necessary to note an observation made in [50]: there has been an interesting trend in the literature with algorithms' performance in their own curated data. In the benchmarking study done in [50] most algorithms performed very well in the yeast datasets but performed poorly in others(human, fly, etc). One reason for this may be that most of these algorithms used the yeast dataset for testing their algorithm's performance in the design phase. These algorithms' performance on different datasets shows that they were mostly over-fitted to the yeast database. The only algorithm with more consistency between the yeast dataset and the other datasets was Weeder, although its performance on other datasets was still inferior. Weeder showed consistent performance in the Metazoan compendium as well and remains one of the best algorithms in the literature.

Given the above it would be interesting to see Amadeus' performance on some other newer benchmarking suites. Also, as shown, Trawler and YMF had good results on some datasets as well. This makes it very hard to compare motif finders since as seen in the convergence graphs of our algorithm, each dataset introduces a completely different solution space and the behavior of each algorithm may be unpredictable. This and the fact that there is no tool that consistently scores better in every dataset are the reasons that this problem has remained open for so long and more motif finders are

introduced every year.

Another problem in the motif discovery field is the question that arises after each run of the algorithm. Are the algorithm's found patterns insignificant or are they maybe a motif that has not yet been discovered? The answer to this question can only be given after actual biological experiments are carried out. This uncertainty adds to the complexity of the problem.

Overall, our algorithm introduces one of the few programs with an evolutionary technique for solving this problem. Its performance is comparable to the best motif finders in the literature. The algorithm found the JASPAR biological motif in 4 datasets out of the 9 analyzed and for the rest it produced motifs for which the biological significance could not be tested. It also showed a possible application in composite motif discovery where transcription factors are analyzed for their coordination with other transcription factors.

9.2 Future Work

The motif discovery problem remains an open and hard problem in bioinformatics. For future work there are many aspects of the algorithm that could be improved:

- **Post-processing stage:** Most motif finders in the literature have a few stages through which each candidate solution goes through some form of optimization. Our current algorithm although able to locate the motif in the archive needs to optimize each of the now candidate motifs PWM's using another machine learning technique like expectation maximization. This way by coupling the power of the GA search and the optimization capacity of EM which is no longer liable to local maxima, the found motifs could be clustered and better fitted to the dataset.
- **Other objectives:** There are many objectives that could be used to distinguish motifs. One of the properties of motifs are their complexity. For example the addition of an objective that measures the pattern complexity could greatly help the algorithm.
- **Extracting motif information:** A complication faced in this work was the ambiguity of the biological significance of a found pattern. As stated in section 8.4, although the algorithm was not able to find the

expected motif for the ETS1_Hollenhorst dataset it found the pattern of CREB_Zhang's expected motif in it. This discovery was only made because the pattern was familiar to the researcher. This process should be automated to test the biological significance of patterns found at the end of each run. This would provide more accurate performance feedback on the algorithm. This process can be automated using web services provided with the commercial version of TRANSFAC. The open source JASPAR database currently does not provide this service.

- Gapped motifs: By manipulation of the NFS-SEM to allow transitions to all states after the current state, we would be able to allow gaps or variable length motifs to be found as well which would be a valuable addition to the algorithm.
- Post-processing general SEMs to figure out other fixed topologies to extract the motif. These may evolve NFS-SEMs, e.g. split-next-first with more than one single track for the pattern. Also, new approaches may create entirely new schemes which would not restrict the general SEM like NFS-SEMs do.

Bibliography

- [1] *Biological Sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids / R. Durbin... [et al.]*. Cambridge, Inlaterra : Cambridge University, 2004.
- [2] Daniel A. Ashlock and Andrew McEachern. Nearest neighbor training of side effect machines for sequence classification. In *CIBCB'10*, 2010.
- [3] Daniel A. Ashlock and Elisabeth Warner. Side effect machines for sequence classification. In *CCECE'08*, 2008.
- [4] Wendy Ashlock and Suprakash Datta. Detecting retroviruses using reading frame information and side effect machines. In *CIBCB'10*, 2010.
- [5] Timothy L. Bailey, Michael E. Baker, and Charles P. Elkan. An artificial intelligence approach to motif discovery in protein sequences: Application to steroid dehydrogenases. *The Journal of Steroid Biochemistry and Molecular Biology*, 62(1):29 – 44, 1997.
- [6] Peter J. Bentley and Jonathan P. Wakefield. Finding acceptable solutions in the pareto-optimal range using multiobjective genetic algorithms. In P.K. Chawdhry, R. Roy, and R.K. Pant, editors, *Soft Computing in Engineering Design and Manufacturing*, pages 231–240. Springer-Verlag, January 1998.
- [7] Peter J. Bentley and Jonathan P. Wakefield. Finding acceptable solutions in the pareto-optimal range using multiobjective genetic algorithms. In *Soft Computing in Engineering Design and Manufacturing*, pages 231–240. Springer-Verlag, January 1998.
- [8] Steven Bergen and Brian J. Ross. Evolutionary art using summed multi-objective ranks. In Rick Riolo, Trent McConaghy, and Ekate-

- rina Vladislavleva, editors, *Genetic Programming Theory and Practice VIII*, volume 8 of *Genetic and Evolutionary Computation*, pages 227–244. Springer New York, 2011.
- [9] J.C. Bryne, E. Valen, M.H.E. Tang, T. Marstrand, O. Winther, I. Da Piedade, A. Krogh, B. Lenhard, and A. Sandelin. Jaspar, the open access database of transcription factor-binding profiles: new content and tools in the 2008 update. *Nucleic acids research*, 36(suppl 1):D102–D106, 2008. n/a.
- [10] Dongsheng Che, Haibo Zhao, and Yinglei Song. Mdga: motif discovery using a genetic algorithm. In *GECCO '05*, pages 447–452, 2005.
- [11] Clare Bates Congdon, Joseph C. Aman, Gerardo M. Nava, H. Rex Gaskins, and Carolyn J. Mattingly. An evaluation of information content as a metric for the inference of putative conserved noncoding regions in DNA sequences using a genetic algorithms approach. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 5:1–14, January 2008.
- [12] Clare Bates Congdon et al. Preliminary results for gami: A genetic algorithms approach to motif inference. In *CIBCB'05*, pages 97–104, 2005.
- [13] Chandonia JM Brenner SE Crooks GE, Hon G. Weblogo: A sequence logo generator. *Genome Research*, (14):1188–1190, 2004.
- [14] Patrik D’haeseleer. How does DNA sequence motif discovery work? *Nature Publishing Group*, 2006.
- [15] V. Batagelj *et. al.* Comparison of distance indices between partitions, part i. In *Data, Science, and Classification*, pages 21–28. Springer Verlag, New York, 2006.
- [16] Patricia A. Evans, Andrew D. Smith, and H. Todd Wareham. On the complexity of finding common approximate substrings. *Theor. Comput. Sci.*, 306(1-3):407–430, September 2003.
- [17] E. Fix and J. L. Hodges. Discriminatory analysis—nonparametric discrimination; consistency properties. Technical Report Project 21-49-004, Report No. 4, USAF School of Aviation Medicine, 1951.

- [18] Dario Floreano and Claudio Mattiussi. *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies*. The MIT Press, 2008.
- [19] Carlos M. Fonseca and Peter J. Fleming. Genetic algorithms for multiobjective optimization: Formulation discussion and generalization. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 416–423, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.
- [20] D. E. Goldberg. *Genetic algorithms in search, optimization and machine learning*. 1989.
- [21] G Z Hertz and G D Stormo. Identifying dna and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, 15(7):563–577, 1999.
- [22] J. H. Holland. Genetic Algorithms Computer programs that” evolve” in ways that resemble natural selection can solve complex problems even their creators do not fully understand. *Scientific American*, 267:66–72, 1992.
- [23] Jeffrey Horn, Nicholas Nafpliotis, and David Goldberg. A niched Pareto genetic algorithm for multiobjective optimization. In *First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, Volume 1*, 1994.
- [24] David W. Deerfield II Hugh B. Nicholas Jr., Alexander J. Ropelewski. Strategies for multiple sequence alignment. *BioTechniques*, 32:572–591, 2002.
- [25] Jason D Hughes, Preston W Estep, Saeed Tavazoie, and George M Church. Computational identification of cis-regulatory elements associated with groups of functionally related genes in *saccharomyces cerevisiae*. *Journal of Molecular Biology*, 296(5):1205 – 1214, 2000.
- [26] Guido H. Jajamovich, Xiaodong Wang, Adam P. Arkin, and Michael S. Samoilov. Bayesian multiple-instance motif discovery with bambi: inference of recombinase and transcription factor binding sites. *Nucleic Acids Research*, 2011.

- [27] Mehmet Kaya. Mogamod: Multi-objective genetic algorithm for motif discovery. *Expert Syst. Appl.*, 36:1039–1047, March 2009.
- [28] Carsten Kemena and Cedric Notredame. Upcoming challenges for multiple sequence alignment methods in the high-throughput era. *Bioinformatics*, 25(19):2455–2465, 2009.
- [29] L J Korn, C L Queen, and M N Wegman. Computer analysis of nucleic acid regulatory sequences. *Proceedings of the National Academy of Sciences*, 74(10):4401–4405, 1977.
- [30] Chaim Linhart, Yonit Halperin, and Ron Shamir. Transcription factor and microRNA motif discovery: The Amadeus platform and a compendium of metazoan target sets. *Genome Research*, 18(7):1180–1189, July 2008.
- [31] Falcon F. M. Liu, Jeffrey J. P. Tsai, R. M. Chen, S. N. Chen, and S. H. Shih. Fmga: Finding motifs by genetic algorithm. *BIBE'04*, page 459, 2004.
- [32] Michael A Lones and Andy M Tyrrell. Regulatory motif discovery using a population clustering evolutionary algorithm. *IEEE ACM Transactions on Computational Biology and Bioinformatics*, 4(3):403–414, 2007.
- [33] Cedric Notredame and Desmond G. Higgins. Saga: Sequence alignment by genetic algorithm. *Nucleic Acids Research*, 24(8):1515–1524, 1996.
- [34] Giulio Pavesi, Paolo Mereghetti, Giancarlo Mauri, and Graziano Pesole. Weeder web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes. *Nucleic Acids Research*, 32(suppl 2):W199–W203, 2004.
- [35] Pavel A. Pevzner and Sing-Hoi Sze. Combinatorial approaches to finding subtle signals in DNA sequences. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 269–278. AAAI Press, 2000.
- [36] Cary Queen, Mark N. Wegman, and Laurence Jay Korn. Improvements to a program for dna analysis: a procedure to find homologies among many sequences. *Nucleic Acids Research*, 10(1):449–456, 1982.

- [37] Emma Redhead and Timothy Bailey. Discriminative motif discovery in DNA and protein sequences using the deme algorithm. *BMC Bioinformatics*, 8:1–19, 2007.
- [38] Brian J. Ross. Evolution of stochastic bio-networks using summed rank strategies. In *IEEE Congress on Evolutionary Computation*, pages 773–780, 2011.
- [39] Olivier Sand, Morgane Thomas-Chollier, and Jacques van Helden. Retrieve-ensembl-seq: user-friendly and large-scale retrieval of single or multi-genome sequences from ensembl. *Bioinformatics*, 25(20):2739–2740, 2009.
- [40] Drablos Finn Sandve Geir. A survey of motif discovery methods in an integrated framework. *Biology Direct*, 1:11, 2006.
- [41] Robert Schleif. In *Genetics and Molecular Biology*, page 23. The Johns Hopkins University Press, 1993.
- [42] T. D. Schneider and R. M. Stephens. Sequence logos: A new way to display consensus sequences. *Nucleic Acids Res.*, 18:6097–6100, 1990.
- [43] Saurabh Sinha and Martin Tompa. Ymf: a program for discovery of novel transcription factor binding sites by statistical overrepresentation. *Nucleic Acids Research*, 31(13):3586–3588, 2003.
- [44] Giulietta Spudich, Xose M. Fernandez-Suarez, and Ewan Birney. Genome browsing with ensembl: a practical overview. *Briefings in Functional Genomics & Proteomics*, 6(3):202–219, 2007.
- [45] N. Srinivas and Kalyanmoy Deb. Multiobjective optimization using non-dominated sorting in genetic algorithms. 1994.
- [46] Rodger Staden. Computer methods to locate signals in nucleic acid sequences. *Nucleic Acids Research*, 12(1Part2):505–519, 1984.
- [47] G D Stormo. Computer methods for analyzing sequence recognition of nucleic acids. *Annual Review of Biophysics and Biophysical Chemistry*, 17(1):241–263, 1988.

- [48] Gary D. Stormo, Thomas D. Schneider, Larry Gold, and Andrzej Ehrenfeucht. Use of the “perceptron” algorithm to distinguish translational initiation sites in *e. coli*. 1982.
- [49] William Thompson, Eric C. Rouchka, and Charles E. Lawrence. Gibbs recursive sampler: finding transcription factor binding sites. *Nucleic Acids Research*, 31(13):3580–3585, 2003.
- [50] Martin Tompa et al. Assessing computational tools for the discovery of transcription factor binding sites. *Nat Biotech*, 23(1):137–144, January 2005.
- [51] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8:173–195, 2000.

Appendix A

Found Motifs

A.1

Figure A.1: Instances of motifs found in the Human.CREB dataset using the NSGA-II approach

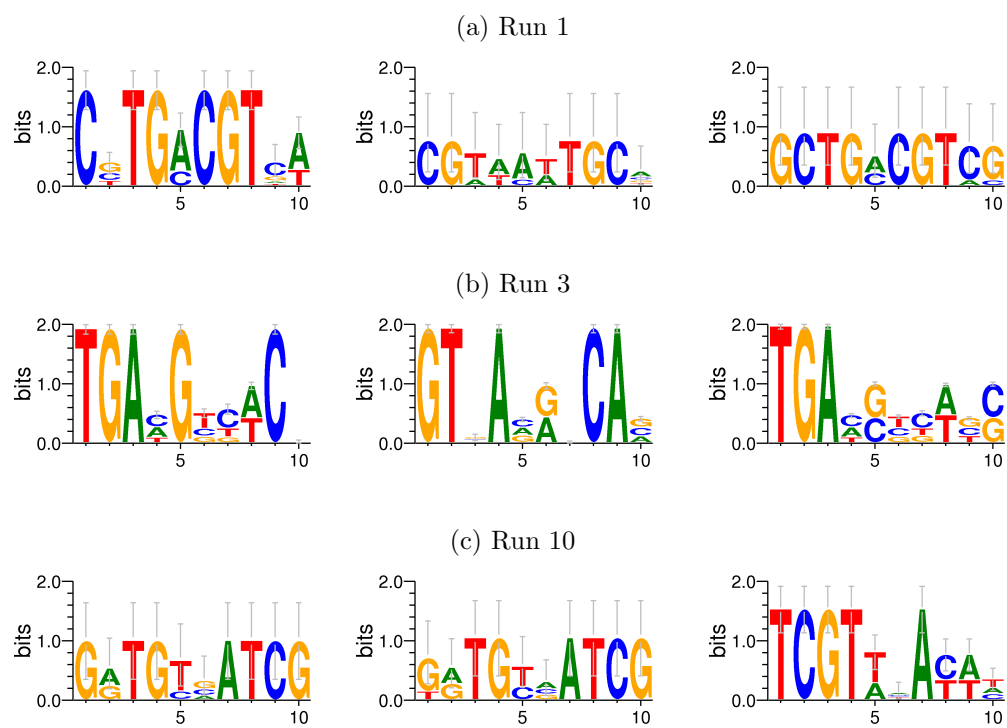


Figure A.2: Instances of motifs found in the *Elegans_GATA* dataset using the NSGA-II approach

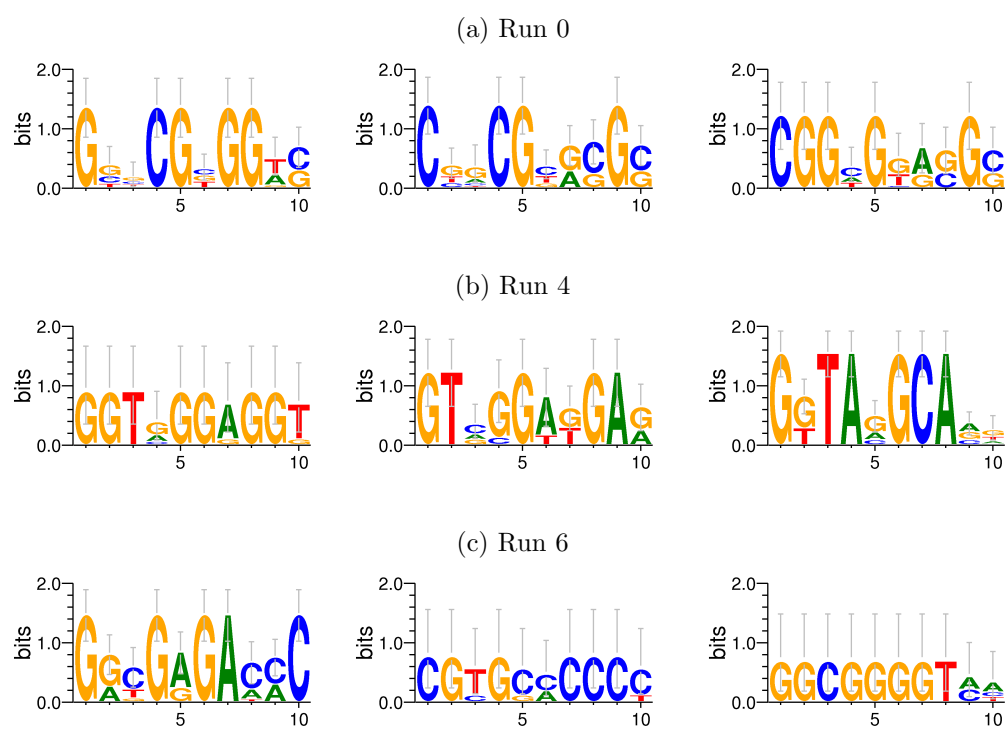


Figure A.3: Instances of motifs found in the Human_HSF1 dataset using the NSGA-II approach

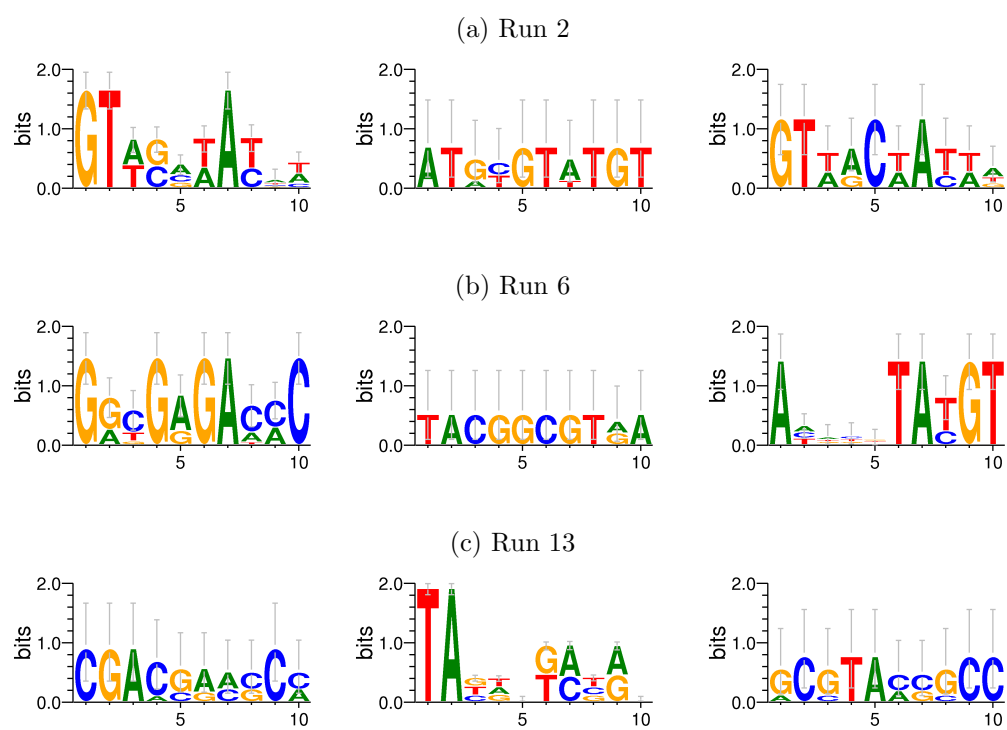


Figure A.4: Instances of motifs found in the Human.CREB dataset using the SPEA2 approach

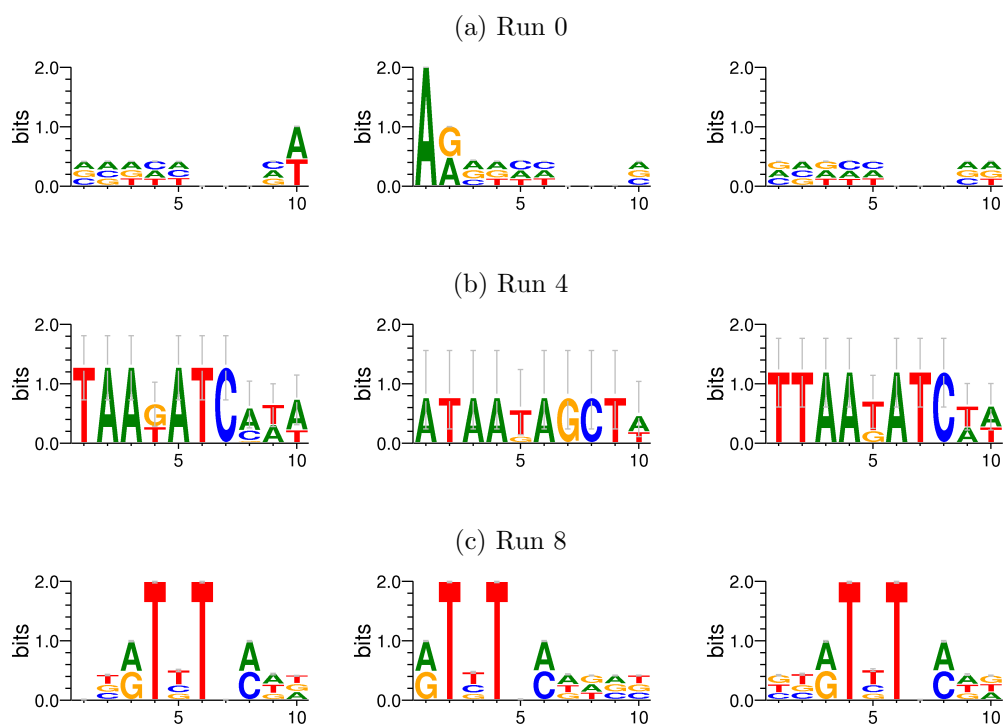


Figure A.5: Instances of motifs found in the Human_GATA dataset using the SPEA2 approach

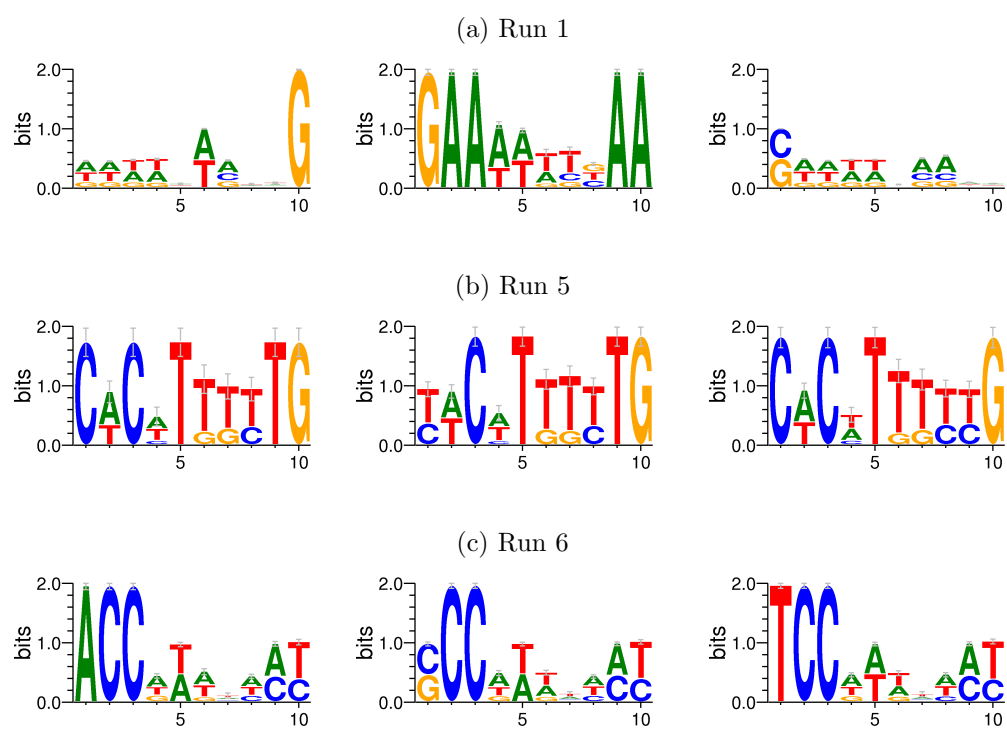


Figure A.6: Instances of motifs found in the Human_HSF1 dataset using the SPEA2 approach

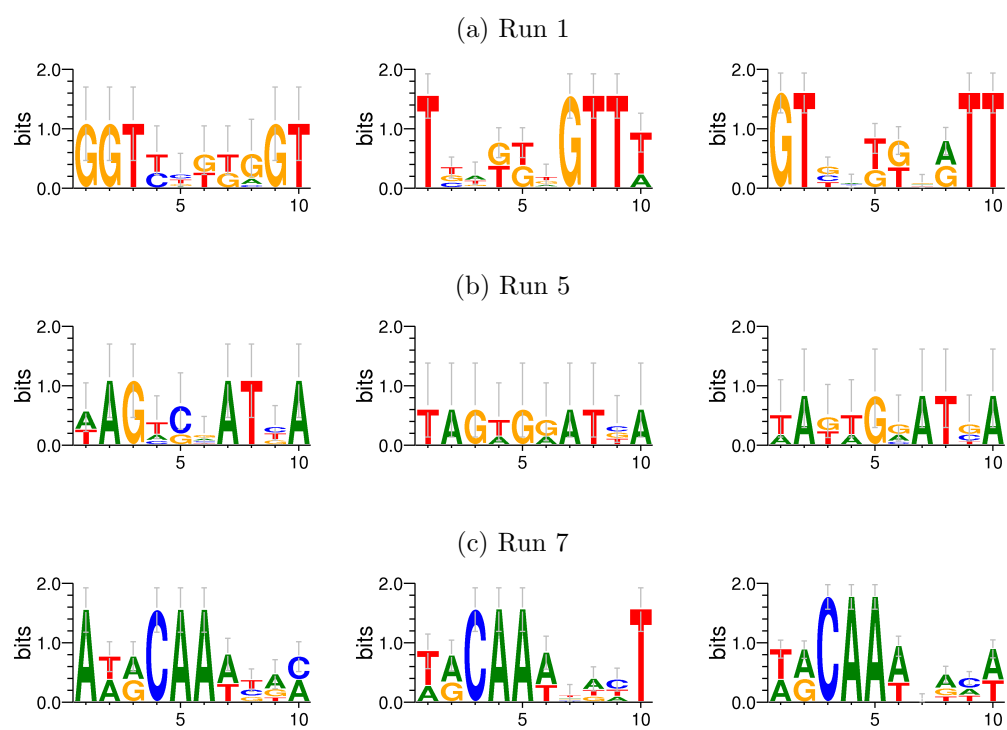


Figure A.7: Instances of motifs found in the Human.CREB dataset using the Summed Ranks approach

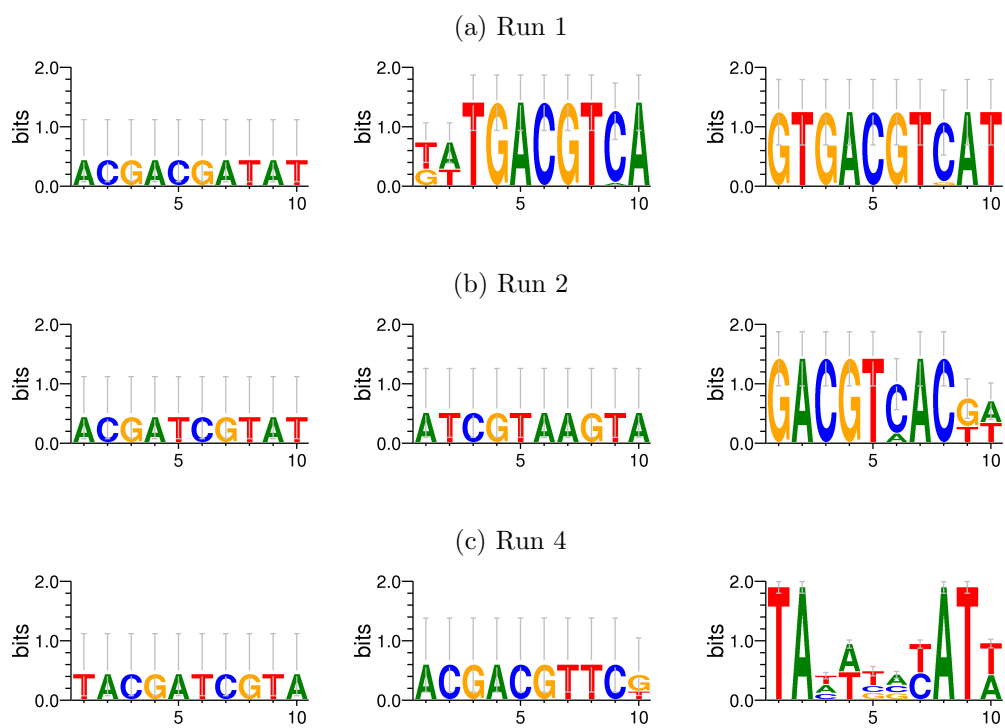


Figure A.8: Instances of motifs found in the *Elegans_GATA* dataset using the Summed Ranks approach

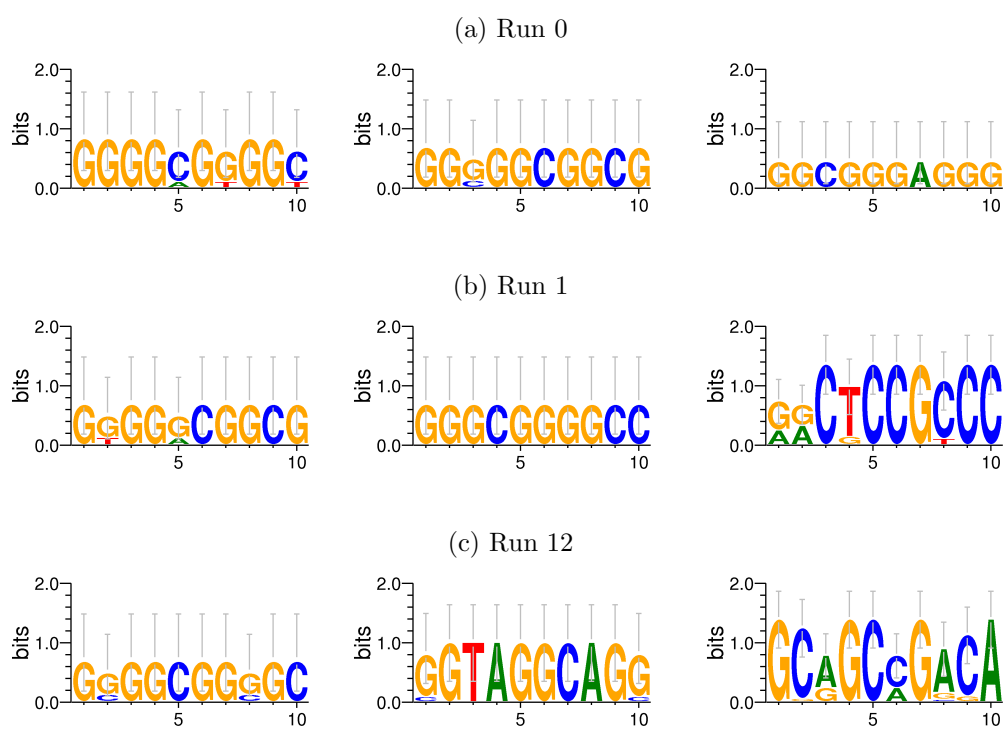
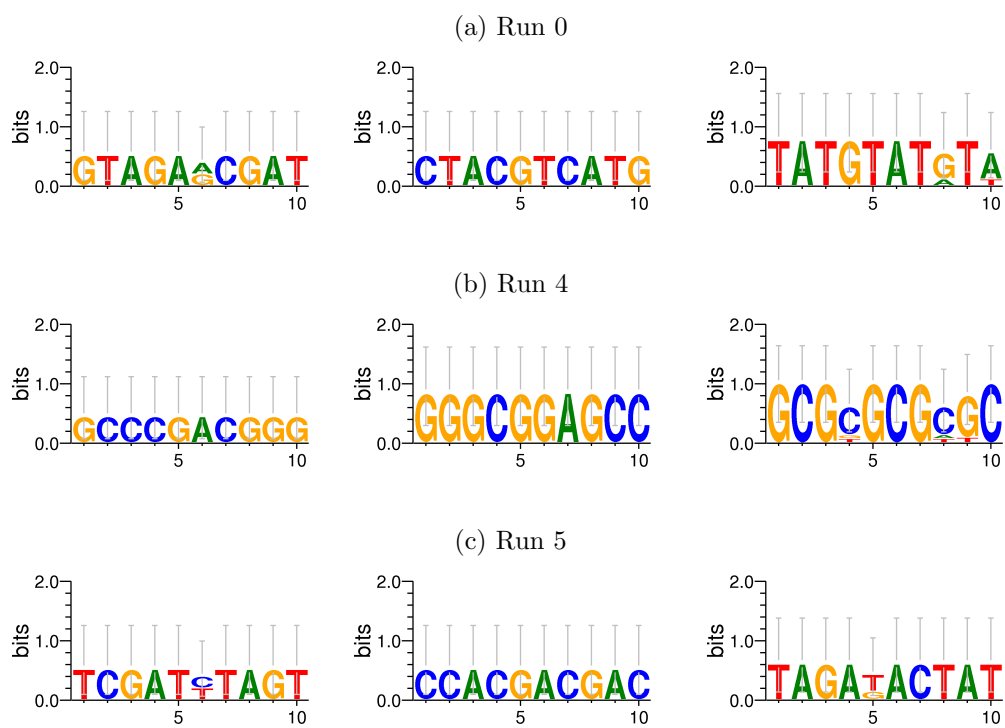


Figure A.9: Instances of motifs found in the Human_HSF1 dataset using the Summed Ranks approach



Appendix B

Convergence Charts

B.1

Figure B.1: Convergence Charts for HNF1a_Odom. The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood

(a) HNF1a_Odom

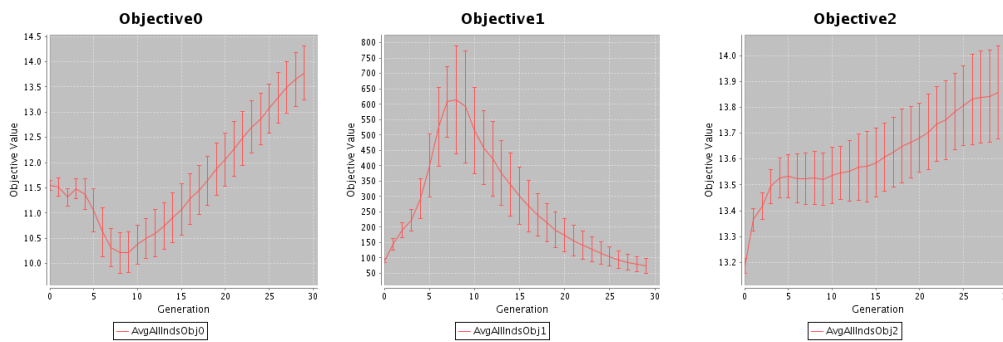


Figure B.2: Convergence Charts for E2F_Ren. The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood

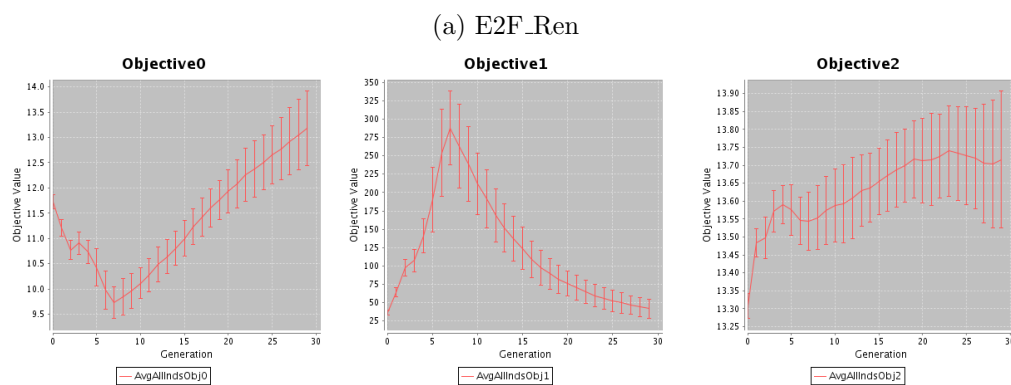


Figure B.3: Convergence Charts for HSF1_Page. The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood

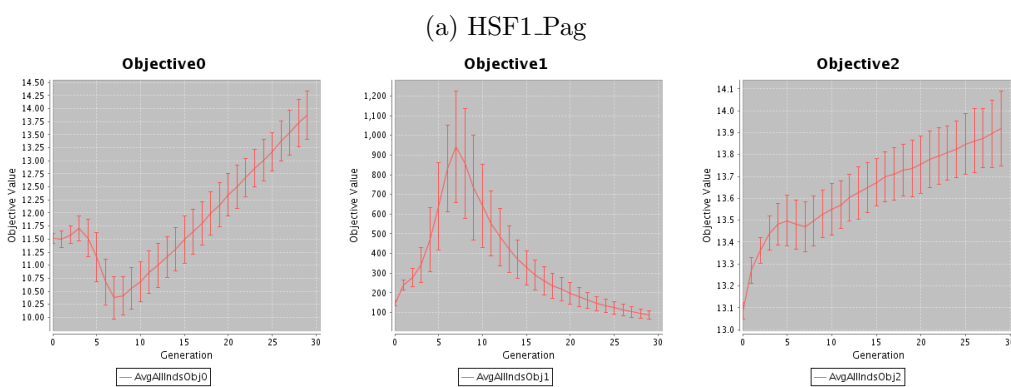


Figure B.4: Myc_Oryan The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood

(a) Myc_Oryan

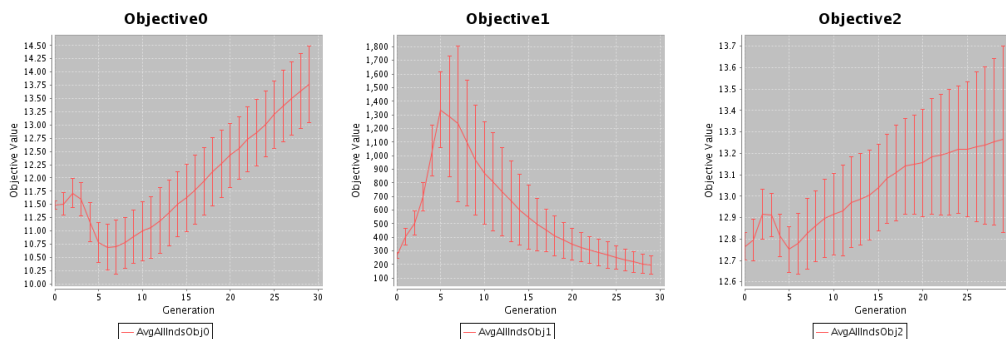


Figure B.5: ETS1_Hollenhorst The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood

(a) ETS1_Hollenhorst

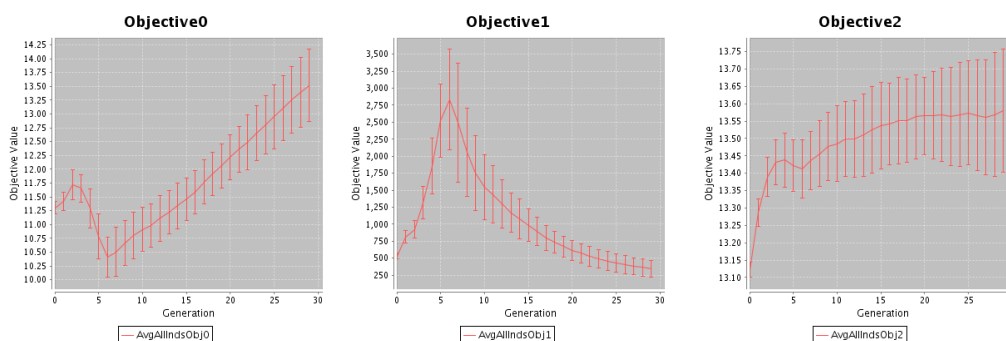


Figure B.6: MEF2_Blais The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood

(a) MEF2_Blais

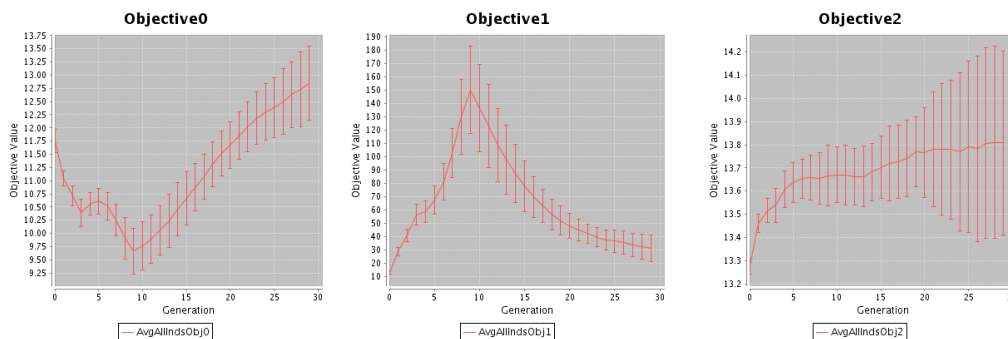


Figure B.7: SRF_Cooper The objectives are as follows : Objective0 = Entropy, Objective1 = Found Motif Count and Objective2 = Likelihood

(a) SRF_Cooper

