San Jose State University
SJSU ScholarWorks

Master's Projects | Master's Theses and Graduate Research

1-1-2008

Motif Discovery in Biological Sequences

Medha Pradhan
San Jose State University

Follow this and additional works at: http://scholarworks.sjsu.edu/etd_projects

Recommended Citation
Pradhan, Medha, "Motif Discovery in Biological Sequences" (2008). *Master's Projects.* Paper 106.

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

Motif Discovery in Biological Sequences

A Project Report

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Computer Science

By

Medha Pradhan

December 2008

© 2008

Medha Pradhan

ALL RIGHTS RESERVED

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Sami Khuri

Dr. Augustin Araya

Dr. Robert Chun

APPROVED FOR THE UNIVERSITY

Abstract

MOTIF DISCOVERY IN BIOLOGICAL SEQUENCES

By Medha Pradhan

With the large amount of biological data generated due to DNA sequencing of various organisms, it is becoming necessary to identify techniques that can help in finding useful information amongst all the data. Finding motifs involves determining meaningful short sequences that may be repeated over many sequences in various species. Various approaches for the motif discovery problem have been proposed in the literature. One method suggests using genetic algorithms. In this project, an evolutionary approach for motif discovery has been explored. The population is clustered during every generation of the algorithm and then evolved locally within the clusters to allow the search space to maintain solution diversity.

ACKNOWLEDGEMENTS

I would like to express my heartfelt appreciation for Dr. Sami Khuri for being my advisor and mentor. His motivation, suggestions and insights for this project have been invaluable. Without his support and guidance this project would not have been possible.

I would also like to extend my gratitude to my committee members, Dr. Augustin Araya and Dr. Robert Chun, for their time and input provided during the work.

Last, but definitely not the least, I would like to thank my husband Salil for his encouragement and help, and for bearing with me for the whole year while I worked on the project.

TABLE OF CONTENTS

Abstract ..... **iii**
ACKNOWLEDGEMENTS ..... **iv**
TABLE OF CONTENTS ..... **v**
LIST OF FIGURES ..... **vi**
LIST OF TABLES ..... **vii**
1 Introduction **1**
2 Background **1**
2.1 Meaning of motifs 1
2.2 Current Techniques for Motif Discovery 3
2.3 Evolutionary approach for motif-discovery 7
2.3.1 Genetic Algorithms *7*
2.3.2 Genetic Algorithms and Motif-Discovery *9*
3 Materials and Methods **9**
3.1 Objective 9
3.2 Approach 10
3.2.1 Representing Motifs *10*
3.2.2 Evaluating Motifs *11*
3.2.3 Clustering the Population *11*
3.2.3.1 Leader Algorithm 12
3.2.3.2 Improved Leader Algorithm 13
3.2.3.3 Improved Leader Algorithm for Motif Discovery 13
3.2.4 Genetic Algorithm for Motif Discovery *16*
4 Data Sets Used **21**
5 Results **23**
6 Evaluation **27**
7 Conclusion **29**
8 References **31**
Appendix A **33**

LIST OF FIGURES

Figure 1. Representing motifs based on frequency 2
Figure 2. Flowchart for a basic genetic algorithm 7
Figure 3. A simple example depicting the crossover process 8
Figure 4. A simple example depicting the mutation process 8
Figure 5. Motif representation being used in the project 10
Figure 6. Converting a Position Frequency Matrix (PFM) for a motif to its corresponding Position Weight Matrix (PWM) 11
Figure 7. Flowchart of the clustering steps needed for clustering the population 15
Figure 8. An example of the mutation1 process 18
Figure 9. An example of the mutation2 process showing a column being inserted at the selected position. 18
Figure 10. An example showing the Uniform Crossover method. 19
Figure 11. Flowchart of the genetic algorithm being used for motif discovery 20

LIST OF TABLES

Table 1. Various parameters used in the genetic algorithm 21
Table 2. Different types of motifs embedded in the synthetic test data 22
Table 3. Findings for test runs on HFH-1 embedded synthetic test data 24
Table 4. Findings for test runs on HLF embedded synthetic test data 24
Table 5. Findings for test runs on multiple motifs embedded synthetic test data 25
Table 6. Findings for test runs on liver-specific data set 26
Table 7. Findings for test runs on muscle-specific data set 27

1 Introduction

In case of biological sequences, motifs can be described as short conserved sequences consisting of patterns of amino acids or bases of nucleotides which can be biologically meaningful. Some of the features of motifs are

- They are patterns of length 10 to 25 bases, and are repeated over many sequences
- They are statistically over-represented in regulatory regions
- They are small, have constant size, and are repeated very often.

Identification of motifs is becoming very important because they represent conserved sequences which can be biologically meaningful. Some of the areas where motif discovery can be useful include finding binding sites in amino acids, finding regulatory information within either DNA or RNA sequences, searching for splicing information, and protein domains. The motifs can represent patterns which activate or inhibit the transcription process and are responsible for regulating gene expression. Motif identification can be thought of as finding the best local multiple alignment for the sequences under consideration.

The challenges present in motif identification include:

- The motifs are never exactly the same as the actual conserved sequence. There is always a lot of sequence variability present with respect to a single motif.
- Motifs are very short signals as compared to the size of the DNA sequence under consideration.
- The regulatory sequences containing the motifs may sometimes be located very far away from coding regions that they regulate. This makes it difficult to determine the portion of the DNA sequence that should be analyzed.
- The regulatory sequences may, at times, be present on the opposite strand from the coding sequence they regulate.
- The motif discovery problem is an NP-Complete problem [4], so there is no polynomial-time solution present for motif discovery.

This report covers an introduction of what motifs are and a short description of the previous work that has been done in this field. It then discusses the evolutionary approach and clustering technique used for finding motifs. The Results section describes the results obtained for various data sets, followed by an Evaluation section which discusses the results in detail.

2 Background

2.1 Meaning of motifs

Motifs are patterns in biological sequences which can indicate the presence of certain biological characteristics. In general, these could represent patterns in any kind of biological sequences such as DNA sequences, RNA sequences, protein sequences etc. For the purpose of this document, a motif will always mean a pattern in a nucleotide sequence, mainly a DNA sequence, which is made up of an alphabet of 4 characters - A,

C, T, G. These represent Adenine, Cytosine, Thymine and Guanine respectively – the bases in the DNA sequences.

Other than the bases representing the coding regions of the genes, the genome consists of a large variety of signals,. These include regulatory signals which aid in promoting or suppressing gene expression, splicing signals that aid in determining which regions of the DNA sequence comprise the coding region, localisation signals which aid in determining where the mature proteins might be located in the cells, and cell cycle signals which help during the cell division process by aiding in recombining and replicating DNA.

Regulatory elements can be broadly divided into two categories, promoters and enhancers. Promoters are normally present close to the first exon and serve as binding sites for the transcription factors. Enhancers, on the other hand, are normally present at a much greater distance from the first exon, and aid in the binding of transcription factors to the promoters. Both these regulatory elements thus work together to bring about expression or suppression of genes. Throughout this document, the motifs under consideration will be those indicating presence of promoter elements. Figure 1 shows the typical representation for a motif.

SAMPLE KNOWN SITES

HEMI3	C	C	C	A	T	T	G	T	T	C	T	C
HEMI3	T	T	T	C	T	G	G	T	T	C	T	C
HEMI3	T	C	A	A	T	T	G	T	T	T	A	G
ANB1	C	T	C	A	T	T	G	T	T	G	T	C
ANB1	T	C	C	A	T	T	G	T	T	C	T	C
ANB1	C	C	A	A	T	T	G	T	T	C	T	C
ANB1	T	C	C	A	T	T	G	T	T	C	G	T
ROX1	C	C	A	A	T	T	G	T	T	T	T	G

FREQUENCY

A	0	0	2	7	0	0	0	0	0	0	1	0
C	4	6	4	1	0	0	0	0	0	5	0	5
T	4	2	2	0	8	7	0	8	8	2	6	1
G	0	0	0	0	0	1	8	0	0	1	1	2

CORRESPONDING MOTIF REPRESENTATION BASED ON FREQUENCY

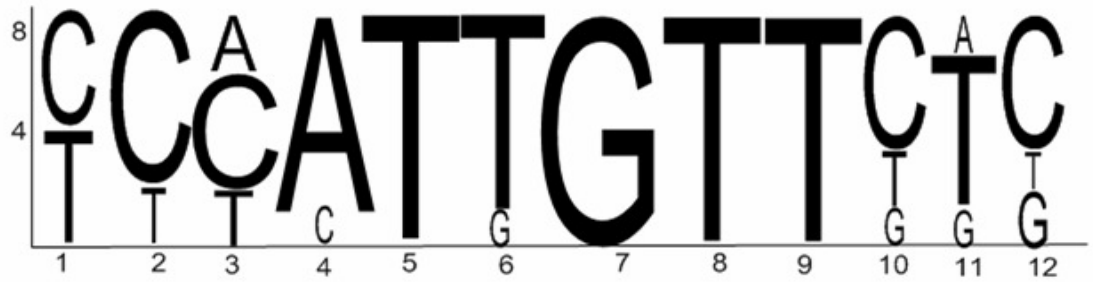

Figure 1. Representing motifs based on frequency
Source: "What are DNA Sequence Motifs", by D'haeseleer, P (2006)

2.2 Current Techniques for Motif Discovery

Because of their ability to identify regulatory regions and other important signals, there has been an increased amount of work going on in the field of motif discovery. Different techniques have been proposed for this. Some of these include:

1. Probabilistic approach

   The probabilistic approach for identifying motifs makes use of various methods such as Gibbs Sampling and Expectation Maximization [9].

   Gibbs Sampling is the more common method used in probabilistic approaches. It involves calculating a combined probability distribution of many random variables. This probability distribution is then used to generate a sequence of samples. The basic idea behind this method involves selecting some substrings at random from the input sequence, and generating position weight matrices (PWM) for these substrings. The process then continues by attempting to replace one of the PWMs by some other randomly selected substring from the same sequence. This replacement is then accepted or rejected on the basis of some pre-calculated score. Some of the motif-discovery algorithms based on Gibbs Sampling include AlignACE, BioProspector, and MotifSampler [9].

   The Expectation Maximization method is also based on a similar concept. This method involves random identification of some initial PWM, and then scanning the set of input sequences o find matches to the PWM. A new PWM is generated on the basis of the identified matches to represent a better motif. This process is then continued until the best match has been found. MEME is a motif-discovery algorithm based on Expectation Maximization [9].

   Scalability is a major limitation for probabilistic approaches. These algorithms have been found to either fail for longer sequences, or show degradation in performance. This has been attributed to the increase in noise with the increase in input sequence length. Other limitations include degradation in performance due to the presence of sub-optimal solutions which may hinder in the process of identifying the actual motif. Also, these methods are based on the assumption that nucleotide positions are independent of each other, which is not true in reality [9].

2. Markov Models

   Markov models have been experimented with extensively for motif discovery. A Markov Model is built by analyzing existing data and building a model that can define well the probability of occurrence of a nucleotide in some position based on the previous nucleotides in the sequence. The order of the model depends on the number of previous elements (nucleotides) that influence the probability of the current position. Developing an accurate and complete fixed-length Markov Model requires availability of a large amount of data for modeling and estimating the

number of parameters. However, since the study of motif discovery is relatively recent, the amount of information present regarding motifs is relatively less. This has proven to be a hindrance in generating accurate Markov Models [21].

Variable Length Markov Models have also been used for motif discovery. These involve developing Markov Models with the assumption that the existence of a nucleotide at some position does not always depend on fixed number of previous positions. This gives the advantage of using less memory and faster computations. However, this method also has the same disadvantage as fixed-length Markov Models, namely, availability of sufficient data for generating the model accurately [21].

3. Exact methods

Although the motif discovery problem is NP-Complete, many exact algorithms have been proposed for identification of motifs. These algorithms, however, can only work on smaller problem instances. They are mainly based on the assumption that the problem instances will contain only up to 20 sequences, with a maximum length of 600 nucleotides [4].

One of the algorithms, *PMSPrune*, based on this concept makes use of the branch-and-bound technique. It works by selecting the first string from the given set of input strings, and generating a complete set of neighbors for every *l*-mer in the string (where *l* is the estimated length of the motif to be identified). The neighbors are generated in a branch-and-bound fashion. A check is done to confirm whether the selected *l*-mer is a motif. This is done by checking whether there are similar *l*-mers in the other sequences of the input set that are close to the identified motif. If not, this process is then continued for the next *l*-mer till the motif is identified [4].

One of the major drawbacks is the exponential increase in the time and memory space requirements with the increase in the size of the motif to be identified. It was found that increasing the size of the motif to be identified by just 4 nucleotides increased the time required for the process by a factor of almost 50. Also, the basic assumption of limiting the number of input sequences to less than 20 and the size of each input sequence to less than 600 has been widely debated over. It is believed by many that such imposed restrictions do not correspond to real life situations where it might be necessary to analyze sequences that might be 1000s of nucleotides long [4].

4. Suffix-trees and box-links

Algorithms using suffix trees and box-links have also been proposed for the motif-discovery problem. Both of them make use of the idea of a *box*, where a *box* is a *k*-tuple of substrings obtained from the input sequences [3].

The suffix-tree concept for motif discovery involves building a suffix-tree for a string, where the branches of the tree correspond to all the suffixes for the string

being considered. A general suffix tree is then built by constructing the tree for each input string consecutively. Each node of the tree stores a Boolean array which keeps a track of sequences from the input set that contain the substring formed by traversing from the root to the node being considered. Once the suffix tree and the Boolean arrays have been formed, then the process of motif extraction starts. This involves traversing the entire tree to find all possible strings of length l that could be potential motifs. For each of these identified strings, a box of length $l-1$ is determined which is at a specified distance from the string under consideration. A jump is made if to such nodes that are within the distance. The information about all the nodes thus jumped to are used to determine whether the current sequence under consideration could be a possible motif. This process is then continued till the complete tree has been traversed to identify the best candidate sequence for the motif [3].

Algorithms based on box-link are similar to the suffix tree. Here, instead of having a Boolean array associated with each node, a box-link is associated with it. A box-link is a data structure that contains information required to traverse between the boxes when navigating the tree [3].

The box-link algorithms were found to be much faster than the suffix-tree approach. However, since the space required to store a box-link information is much higher than a Boolean array, the box-link approach was found to have a much higher space cost associated with storing the box-links for each node [3].

Both these approaches, however, are faced with a limitation that they can find only one motif per sequence. Also, they can only find those motifs that are always at the same distance from the consensus sequence. This puts a huge limitation on the variability of the motifs [3].

5. Random and Uniform Projections

Random and Uniform projections are two more methods that have been explored for motif discovery. The Random Projection technique was developed first, and was later enhanced to build the Uniform Projection technique. Although both these techniques use probabilistic methods in the later stages to enhance the process of motif discovery, they provide very good initial starting points for this process. They are based on creating a model for the motifs by taking into account all those l-mers from the input sequences that have a consensus at k positions. A projection, in such cases, is a string of length k ($k \leq l$), where each character in the string is generated from the set of input sequences as follows: For every substring of length l in S, and k distinct integers such that the integers have a property

$$1 \leq p_1 < p_2 < \ldots. < p_k \leq l,$$

the string $s_{p1}s_{p2}s_{p3}\ldots.s_{pk}$ that can be generated from the substring is called a projection [2], [17].

Random Projections work by generating a projection of length k randomly, as described above. A hash table of size 4^k is then created, which can be indexed by all possible strings of length k. Each entry of the hash table holds a bucket containing a

number of strings. Every substring of length *l* from the set of input sequences is placed in the appropriate bucket in the hash table. This process is continued for all possible substrings. Any bucket containing more than *z* strings (where $z > 0$ and is pre-determined before algorithm execution) is termed as an '*enriched*' bucket. The substrings in the enriched bucket are then used as the initial model for the process of motif discovery. This initial model is gradually improved by applying various possible optimization techniques such as Expectation Maximization (EM) [1], SP-Star algorithm [15] etc. [2].

Uniform projection method, improves this process even further, by increasing efficiency and reducing running times. This is achieved by considering a threshold for all projections, called as *discrepancy value,* which is the hamming distance between the maximum (over all projections) and each individual projection. Only those projections having a low discrepancy value are considered for identifying motifs. This method uses a greedy approach where initially all projections covering all *1*-tuples are considered, based on their discrepancy values. These projections are further enhanced by adding more projections covering all *2*-tuples that have not already been considered. This process is repeated until all necessary tuples are considered. Optimization techniques such as EM are then used to refine this and obtain the motif information [17].

Uniform projections have been found to be better than Random Projections since the random algorithm works by considering the entire *k*-projection space. This tends to increase the running time, thus reducing efficiency. The Uniform algorithm samples a smaller space, based on the discrepancy value, and improves efficiency without affecting the accuracy. However, both these algorithms suffer from a few drawbacks including dependence on the parameters for the projections as well as difficulty in determining the stopping criteria [2], [17].

6. Attempts have also been made at using machine-learning approach to identify motifs. However, this approach has not proved very successful, primarily because the results generated are very rarely biologically meaningful [12].

In conclusion, it can be said that although each of these approaches have their own advantages, they also face a lot of limitations. Probabilistic approaches do not allow for gaps, are not very scalable, and result in degradation of performance with increase in size of the input sequence. Markov Models, although being very useful in other areas of bioinformatics, do not work very well for the motif-discovery problem due to lack of insufficient information. Exact methods are useful only for theoretical purposes. Attempting to use exact algorithms for solving NP-Complete problems is not recommended, since it results in an exponential increase in time as well as space with a small increase in the problem instance size. Suffix-trees and box-links can only identify one motif per sequence, at a fixed distance from the consensus sequence. Random and Uniform projections are largely dependent on the parameters that have to be identified for the projections, as well as suffer from the inability, to focus on a fixed stopping criterion.

2.3 Evolutionary approach for motif-discovery

2.3.1 Genetic Algorithms

A genetic algorithm is an approach based on evolutionary computing, which involves using the concept of evolution (reproduction) in order to evolve the solutions. These solutions are scored based on pre-determined fitness functions which evaluate how fit each solution is, thereby aiding decisions regarding which solution needs to be maintained or discarded. Figure 2 shows the flow for a basic genetic algorithm.

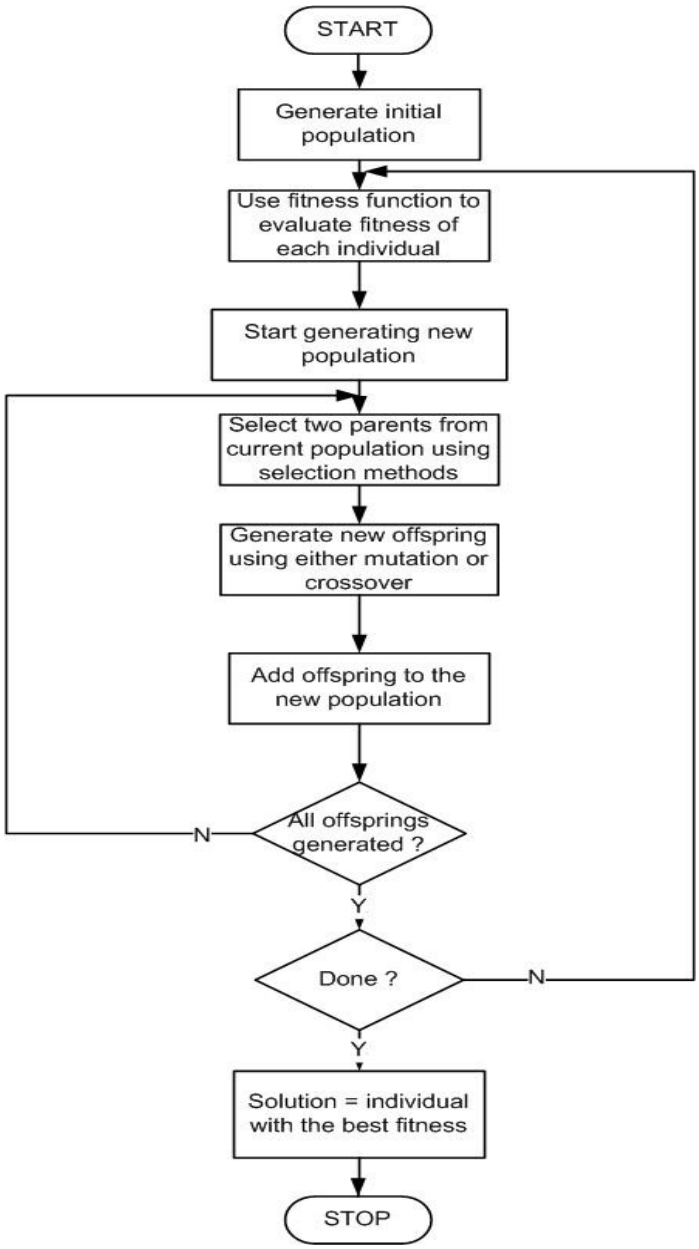

Figure 2. Flowchart for a basic genetic algorithm

The process of generating new offspring is mainly done using two methods, crossover and mutations. The parents are selected using specified selection techniques. The most common technique for parent selection is the roulette-wheel method. In this method, the elements of the population are assigned slots on the roulette wheel, where the size of each slot is proportional to the fitness of the element. The roulette wheel is then spun randomly, and the slot (element) where it stops is chosen as the parent. This method allows the parents to be selected based on the fitness; where more fit solutions have a higher chance of getting selected.

The most basic crossover process is as follows. Two parents are chosen from the population, based on some selection technique. A crossover-site is selected at random, and the right-most strings are swapped, generating two children. Figure 3 depicts an example for the crossover process.

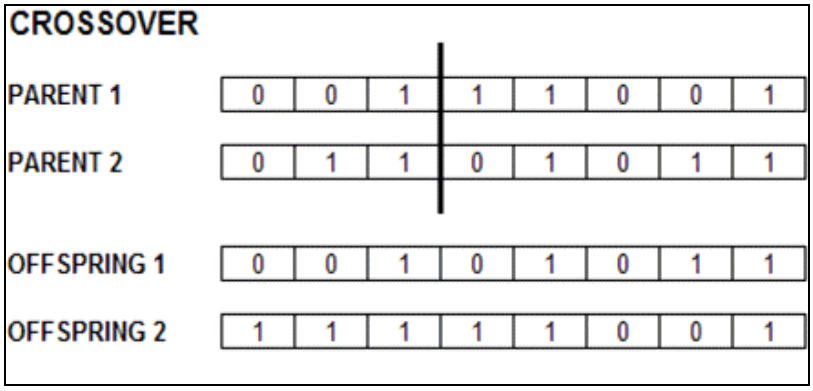

Figure 3. A simple example depicting the crossover process

Mutation works by randomly choosing some position for one of the elements, and changing the value at that position. If the encoding technique for the problem instance is bits, then mutation works by simply flipping the bit at the selected position. Figure 4 depicts an example of how mutation can be performed to generate new offspring.

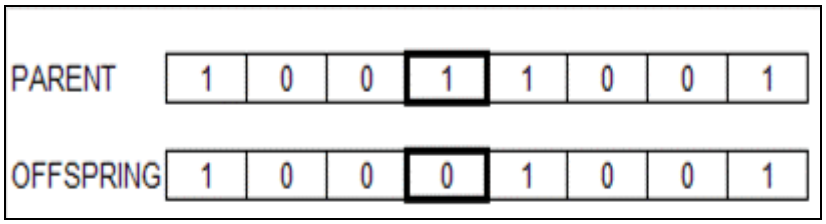

Figure 4. A simple example depicting the mutation process

Genetic algorithms have a large number of advantages. The most important advantage of this is that it is an extremely robust technique. The use of genetic algorithms is not limited to only a particular family of problems. They can be used for solving a wide variety of problems, and have been found to be extremely successful in those areas where the search space is large. Also, although genetic algorithms do not guarantee the best

solution every time, they are very useful in finding acceptable solutions faster than other methods. They have the ability to be hybridized with existing techniques of solving some problems to give better solutions.

However, the general evolutionary approach has some disadvantages as well. Often, they promote the final solution to converge to a local maximum because of some instances of exceptionally high fitness candidates, which are not necessarily optimal. If this occurs, then the algorithm is no longer able to find better solutions because crossovers lead to identical solutions with higher fitness, making it of little use. In most cases, mutation on its own, results in a large reduction in the speed of searching. Thus the overall process becomes very slow, resulting in the need for larger number of generations to be obtained.

2.3.2 Genetic Algorithms and Motif-Discovery

Genetic algorithms have been extensively used in the field of bioinformatics, mainly with respect to multiple sequence alignments (MSA), and have proved quite successful. Recently, various attempts have been made at using the evolutionary approach for motif discovery. This technique, although somewhat similar in concept to machine learning, is much more successful. This is mainly because machine-learning algorithms perform local searches, thus generating solutions that may be locally optimal, but are seldom globally optimal. Genetic algorithms, on the other hand, perform a global search of the search-space without performing an exhaustive search. As a result of this, although the genetic algorithms do not always guarantee an optimal solution, they have a much better chance of finding an optimal solution.

Genetic algorithms also allow using fitness functions for scoring the solutions. These fitness functions need not be constant for all problems; they can use any relevant information to score the solutions including biological information, functional information, etc. Thus, they provide flexibility in evaluating the solutions. Also, the genetic algorithms provide a flexibility of deciding how to represent the problem instance. There are some studies where motifs have been represented in various different formats suitable for the genetic algorithm under consideration. These include representing motifs as regular expressions, position frequency matrices, etc.

There has not yet been as much study in the area of evolutionary approach for motif discovery, as there has been for the other approaches mentioned previously. Some early work focused on using steady-state algorithms for evolving the solution. Recently, a genetic algorithm has been used for evolving not only the content of the motif, but also the position using special crossover operators [12].

3 Materials and Methods

3.1 Objective

The objective of this project is to identify motifs that represent regulatory elements in biological sequences. The input to the algorithm consists of two sets of sequences. The

first is a set of promoter sequences which has the likelihood of containing regulatory elements. The second is a set of background sequences representing some random portion of the DNA which may not include any promoter sequences. The algorithm processes these input sequences to determine regions in the first set which could classify as motifs. This is done by finding subsequences that are over-represented in the first set of input sequences, as compared to the background sequences. These over-represented portions indicate the presence of motifs.

Very few genetic algorithm based approaches make use of such background sequence sets to determine motifs. In most cases, a pre-determined background model is used. The use of such pre-defined background models affects the efficiency of the system, since it does not provide the flexibility of adapting to the input data.

3.2 Approach

The approach being followed in this project for motif-discovery can be summarized as follows:

1. Determine a representation for the motifs.
2. Evaluate the motifs using a well-defined fitness function.
3. Cluster the population based on some clustering metric.
4. Run the genetic algorithm to repeat the above steps until a motif close to the consensus sequence is identified.

3.2.1 Representing Motifs

In general, genetic algorithms provide the flexibility of being able to use a representation that best suits the problem instance and the proposed solutions. In case of motif discovery, various representations have been used previously. These include regular expressions, position frequency matrices, position weight matrices etc.

The representation representation for motifs being used in this project is a combination of both position frequency matrices (PFM) and position weight matrices (PWM). These representations are most frequently used, and have also been found to be the most useful and easy to work with. Figure 5 below shows a brief example of the motif representation being used for this project.

A	0.0	0.3	0.4	1.0	0.0	0.0	0.8	0.2
C	0.4	0.3	0.0	0.0	0.8	0.0	0.0	0.6
T	0.6	0.2	0.0	0.0	0.0	0.8	0.0	0.2
G	0.0	0.2	0.6	0.0	0.2	0.2	0.2	0.0

Sequence logo

Position frequency matrix

Figure 5. Motif representation being used in the project

3.2.2 Evaluating Motifs

In terms of motifs, the fitness can be described as how well it can help in differentiating between the two sets of input sequences, i.e. the first set of promoter sequences and the second set of background sequences.

The fitness function used to evaluate the motifs can be described as follows:

1. The motif under consideration is converted to its representation, i.e. a PFM
2. The PFM is then converted to a PWM by multiplying each element of the PFM by 4, and taking the natural log of this value. Figure 6 shows an example of the PWM values for a sample PFM.
3. Each and every sequence of the input set (set of promoter sequences) and the background set is then considered. The PWM match score for each sequence is calculated as follows:
    a. The PWM match score at each offset of the input sequence is calculated using the PWM generated above.
    b. The maximum match value in the sequence is considered as the best match for the entire sequence.
    c. This value is divided by the maximum possible value for any PWM of the specified size in order to normalize it to a range of [0, 1].
    d. This process is repeated for every sequence in both data sets.
4. The average best match value is then calculated for each of the two data sets.
5. The difference between the two average values, calculated as follows, gives the fitness of the motif

    average best match score for the promoter sequence - *average best match score for the background sequence*

    A higher positive value indicates that the motif is represented much better in the set of promoter sequences than the background sequences.

PFM

A	0.0	0.4	0.4	1.0	0.0	0.0	0.8	0.2
C	0.4	0.2	0.0	0.0	0.8	0.0	0.0	0.0
T	0.6	0.2	0.0	0.0	0.0	0.8	0.0	0.4
G	0.0	0.2	0.6	0.0	0.2	0.2	0.2	0.4

PWM

-∞	0.47	0.47	1.38	-∞	-∞	1.16	-0.22
0.47	-0.22	-∞	-∞	1.16	-∞	-∞	-∞
0.87	-0.22	-∞	-∞	-∞	1.16	-∞	0.47
-∞	-0.22	0.87	-∞	-0.22	-0.22	-0.22	0.47

Figure 6. Converting a Position Frequency Matrix (PFM) for a motif to its corresponding Position Weight Matrix (PWM)

3.2.3 Clustering the Population

This section explains in detail the need for clustering during motif discovery, the clustering algorithm used, and customization of the clustering algorithm for the purpose of motif discovery.

Because regulatory elements provide binding sites for transcription factors, it is important to bear in mind that there is always a possibility of the presence of multiple motifs in the regulatory regions of biological sequences. This can pose some problems when considering the standard genetic algorithm approach.

A standard genetic algorithm works by evolving the population such that the fittest solution always represents the best possible candidate solution. Thus, genetic algorithms almost always provide a single solution at the end of the process, or, very similar solutions, in case of multiple solutions. As a result, the diversity in the population is not maintained, leading to the loss of information about the multiple diverse motifs. Another drawback is that many times the single solution obtained turns out to be a false positive. It is possible for some solutions with apparently higher fitness, to be biologically meaningless.

Some techniques have been proposed for overcoming these drawbacks. These include controlling the choice of parents selected for mating, using spatially distributed populations, population clustering etc.

The population clustering technique has been chosen for this project. It involves using a clustering algorithm to divide the population. The clusters are then subject to the mating process in the genetic algorithm, ensuring that mating occurs within the clusters. Such intra-cluster mating allows solution diversity to be maintained during evolution. After this, the new population is again subject to clustering in the new iteration. This provides the advantage of allowing the solution to move from one cluster to another (based of fitness), thus promoting some degree of inter-cluster mating, providing another means of maintaining solution diversity.

3.2.3.1 Leader Algorithm

Leader algorithm, a type of quick-partition algorithm, is one of the classic and oldest clustering techniques present. Quick partition-based algorithms provide a much faster method for clustering the solution space, as compared to other clustering methods. The Leader algorithm is one of the simplest and fastest clustering methods present, and is most appropriate when a large number of objects are to be clustered.

The Leader algorithm is a one-pass algorithm used to partition the space into M partitions by calculating the Euclidean distance between all pairs of objects and comparing it to a threshold T. It constructs partitions equal to the number of clusters. For each cluster a leading solution is selected (leader), such that all elements in the cluster are within a distance T from the leader. It is a greedy approach where only one pass is made through the solution space, and each solution is assigned to the first cluster whose leader is close enough. If none of the leaders are within the distance T, then a new cluster is created, and the solution is made the leader of the cluster [8].

3.2.3.2 Improved Leader Algorithm

For the purpose of this project a modified version of the Leader algorithm, called as Improved Leader Algorithm, has been used. The modified version is a multi-pass method. This method does not require calculation of thresholds. It begins with an initial central object, and on the first pass, finds an object furthest from the central. This object becomes the leader of the next cluster. The next pass involves finding an object that is farthest from its corresponding leader. That object is then made the Leader of the next cluster. This process is repeated until the required number of clusters have been formed, and all objects have been clustered. The number of passes is equal to the number of clusters needed to be formed. This method provides an advantage over the classic Leader algorithm, in that no thresholds are required to be estimated before-hand. But it is necessary to estimate the number of clusters. However, it is easier to estimate the number of clusters as opposed to estimating the thresholds. Also, the classic Leader algorithm requires sorting all the objects in reverse order before they are clustered in order to avoid inflated clusters. The Improved algorithm is invariant under the change of input order [8].

The steps in the Improved Leader Algorithm are shown below. The variables used are: K for the cluster number, I for the elements under consideration, L for the assigned leaders of the cluster, D for the distance between two elements.

Algorithm 1: Improved Leader Algorithm for clustering elements

1. Begin with the central element in I. Place the object in cluster K = 1, and make it the leader L_1 of cluster K = 1.
2. For each element in I, repeat steps 2.1 and 2.2
    a. For each cluster K generated so far, find distance of the element from the leader of the cluster i.e $D(I,L_K)$.
    b. Assign element I to cluster K where $D(I,L_K)$ is minimum.
3. For each cluster K generated so far, find the element that is farthest from its leader i.e. $d_{max} = \max(D(I,L_K))$.
4. Select the element having maximum d_{max} value, and make it the leader of a new cluster K = K + 1 i.e. L_{K+1}
5. Repeat steps 2 to 5 until the required number of clusters are created.

3.2.3.3 Improved Leader Algorithm for Motif Discovery

The improved leader algorithm has been customized for the motif-discovery problem. The distance between two PFMs is measured using a *feature vector*, which calculates the distance based on all the tetra-nucleotide distributions of the PFM under consideration [13]. Previously conducted experiments have shown that using tetra-nucleotide distributions give the best performance during execution, as opposed to bi-nucleotide and tri-nucleotide distributions [13]. The *feature vector* uses the tetra-nucleotide distributions to calculate the sum of probabilities at each offset which is then divided by the motif length to normalize the answer. The distance between any two motifs is then calculated as the Euclidean distance between the corresponding *feature vectors*. This approach for calculating the feature vector was proposed by Lones and Tyrrell in [13].

Algorithm 2 shows the steps for calculating the feature vector. Algorithm 3 describes the steps for calculating the Euclidean distance between two feature vectors. Figure 7 gives the flowchart of the clustering steps used for clustering the population.

Algorithm 2: Steps for calculating the feature vector

1. Let *index* = 0
2. for all tetra-nucleotides in the following set: {AAAA, AAAC, AAAG, …. , TTTG, TTTT}
    a. $prob_{sum}$ = 0, *cnt* = 0
    b. **for** *i* from 0 to (number of columns in the PFM – 4)
        i. *prob* = 1
        ii. *cnt* = *cnt* + 1
        iii. **for** each residue of the tetra-nucleotide under consideration
            1. *prob* = *prob* * residue value of PFM column (*i*+residue position) ## Calculate probability for each residue ##
        iv. **end for**
        v. $prob_{sum} = prob_{sum} + p$ ## Calculate total probability for all tetra-nucleotides ##
    c. **end for**
    d. *featureVector*[*index*] = $prob_{sum}/cnt$ ## normalize ##
    e. *index* = *index* + 1
3. **end for**

Algorithm 3: Calculating the Euclidean distance between two feature vectors

1. *sum* = 0, *i* = 0
2. for each element *i* of the two feature vectors *fv1* and *fv2*
    a. *diff* = fv1[i] – fv2[i]
    b. *sum* = *sum* + (*diff* * *diff*)
3. *euclideanDistance* = sqrt(*sum*)

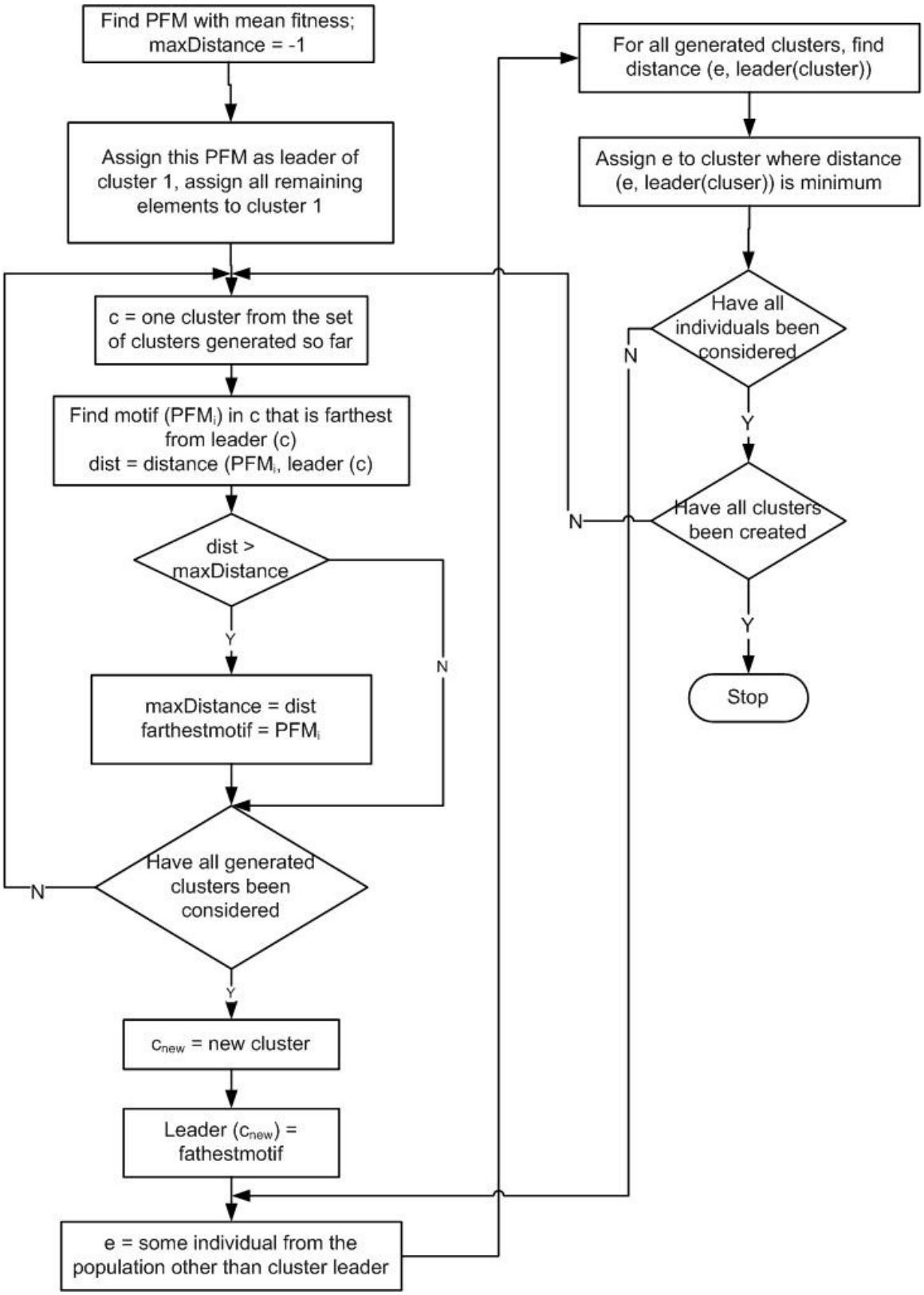

Figure 7. Flowchart of the clustering steps needed for clustering the population

3.2.4 Genetic Algorithm for Motif Discovery

This section gives a detailed explanation of the genetic algorithm used for motif discovery. It describes in detail the selection technique for the population, the various methods used for generating the new population, and a flow of the steps in the genetic algorithm.

The genetic algorithm used for motif discovery starts with the initialization phase. This phase is then followed by an iterative process for clustering the population, mating the selected parents (using elitism, mutation and crossover), and evaluating the new offspring generated.

The initialization phase consists of randomly generating the initial population. The population is evaluated using the fitness function described in section 3.2.2. This population is then clustered using the approach described in section 3.2.3. After clustering has taken place, the population is allowed to evolve through the process of mating. Mating is only allowed within clusters, which helps to maintain the solution diversity. The number of offspring generated for each cluster is proportional to the mean fitness of all the solutions in the cluster. Mutation and crossover are done with a mutation to crossover ratio of 7:3. After the reproduction phase, the entire process is again repeated. The various parameters used in the genetic algorithm are summarized in Table 1. The flowchart for the overall genetic algorithm is shown in Figure 11.

<u>Selection</u>

In order to perform mutations and crossover, the parents are selected using a selection method known as 'Rank Selection'. The roulette wheel method is the most commonly used method for the selection process. However, it does not perform very well when there are large differences between the fitness of the objects in the population. Thus, if there are some objects that have a high fitness like 80%, and others have a much lesser fitness, then these lower fitness objects will have only a very small chance of ever being selected. To overcome this problem, the Rank Selection method is used. This method slightly modifies the roulette wheel method. It ranks all the objects based on fitness (from 1 to N, where N is the population size). The slots in the roulette wheel are then assigned based on these ranks, instead of the fitness. This gives all these object a chance of being selected based on their ranks.

The steps used for the Rank Selection method are:

1. Rank all elements of the cluster on the basis of their fitness, such that the element with the lowest fitness has rank 1, and the one with the highest fitness will have rank N.
   Ex: For 4 elements with fitness 0.3, 0.6, 0.9 and 0.1, their respective ranks will be 2, 3, 4 and 1. For elements with fitness 0.3, 0.6, 0.6 and 0.1, their respective ranks will be 2, 4, 4, and 1.
2. Find the sum of all ranks

3. The percentage of the roulette wheel assigned for each element is equal to
   *(rank*100) / sum of ranks*
4. Spin the roulette wheel.
5. Determine the element corresponding to the slot selected.
6. Select this element

<u>Reproduction</u>

Three methods are used for mating. They are:

1. Elitism

   Elitism implies that the instance with the best fitness value is allowed to propagate to the next generation, without any modifications. This is done because many times the best solution fades away due to mutations and crossovers. Elitism allows the best solution of the population to move ahead as is. If the solution is not good enough, it will eventually get discarded during evolution.

2. Mutation

   As described earlier, mutation is carried out by randomly selecting a position and changing its value. Mutation is carried out by selecting only one parent using the Rank Selection method described above. For the purpose of the project, two types of mutations are performed:
   a. Mutation 1: This mutation is applied with the probability of 10% per nucleotide position of the PFM. Figure 8 shows an example of how this mutation process works.The steps are as follows:
      i. The nucleotide position of the PFM is selected using a 10% probability per position.
      ii. For the selected position, the base (A, C, T, or G) is then randomly selected.
      iii. The frequency of the base is changed by providing a standard deviation within the range of +/-0.5.
      iv. If the frequency of the position exceeds 1, then these changes are discarded and the original value is restored.
      v. The frequencies of the rest of the bases are then again randomly assigned such that the total sum of frequencies for all four bases of the selected nucleotide position does not exceed 1.
      vi. Add this offspring to the new population.

PARENT	0.2	0.6	0.8	0.0	0.2	0.0
	0.2	0.0	0.0	0.6	0.4	0.0
	0.4	0.0	0.2	0.2	0.4	0.0
	0.2	0.4	0.0	0.2	0.0	1.0
OFFSPRING	0.2	0.6	0.4	0.0	0.2	0.0
	0.2	0.0	0.4	0.6	0.4	0.0
	0.4	0.0	0.0	0.2	0.4	0.0
	0.2	0.4	0.2	0.2	0.0	1.0

Figure 8. An example of the mutation1 process
Here the cell shown in yellow is mutated by a change in frequency of 0.4. The remaining cells for that position are then randomly changed, ensuring that the values sum to unity.

b. Mutation 2: This mutation is performed very rarely, with a likelihood of being applied only 4% of the times. It involves randomly adding a new column in the PFM under consideration. This causes the size of the motif under consideration to increase by 1. Figure 9 shows an example for this mutation process. The steps are as follows:
    i. Randomly select the position at which the new column has to be added.
    ii. Copy all values of the PFM under consideration, up to the position number generated, to the child.
    iii. At the selected position, randomly generate the frequencies for the four bases. Ensure that the sum of the bases for the selected position does not exceed 1.
    iv. Copy all the remaining values from the original PFM into the new offspring.
    v. Add this offspring to the new population.

MUTATION 2

selected position

PARENT

0.2	0.6	0.8	0.0	0.2	0.0
0.2	0.0	0.0	0.6	0.4	0.0
0.4	0.0	0.2	0.2	0.4	0.0
0.2	0.4	0.0	0.2	0.0	1.0

OFFSPRING

0.2	0.6	0.8	0.0	0.2	0.0	0.0
0.2	0.0	0.0	0.6	0.4	0.6	0.0
0.4	0.0	0.2	0.2	0.4	0.0	0.0
0.2	0.4	0.0	0.2	0.0	0.4	1.0

Figure 9. An example of the mutation2 process showing a column being inserted at the selected position.

3. Crossover

As described in the earlier section, crossover means selecting two parents and interchanging data from them to generate new solutions. Under normal circumstances, crossover results in the generation of two offspring. The crossover mechanism used for this project, however, generates only one offspring. The technique used is called Uniform Crossover. In Uniform crossover, the value at each position of the offspring is derived from either of the parents, normally with a 50% probability of selecting either parent. In some cases, the percentage probability of selecting the position data can be different, depending on whether parents with higher probability need to be favored. Figure 10 shows an example of how the Uniform Crossover method works for this project.

The Uniform Crossover method is applied as follows for this project:

a. Select two parents using the Rank Selection method.
b. Let the size of the offspring be equal to the size of the larger parent.
c. For each position of the offspring, select the parent whose values will be used with a 50% probability of selecting either parent.
d. Copy the frequency values of the selected parent for the position being considered.
e. Repeat steps (c) and (d) until the number of positions covered = size of the smaller parent.
f. Copy values for the remaining positions from the larger parent.
g. Add this offspring to the new population.

PARENT 1

0.2	0.6	0.8	0.0	0.2	0.0
0.2	0.0	0.0	0.6	0.4	0.0
0.4	0.0	0.2	0.2	0.4	0.0
0.2	0.4	0.0	0.2	0.0	1.0

PARENT 2

0.0	0.8	0.8	0.2	0.0	0.2
0.6	0.0	0.2	0.2	0.2	0.0
0.4	0.0	0.0	0.6	0.2	0.0
0.0	0.2	0.0	0.0	0.6	0.8

OFFSPRING

0.2	0.6	0.8	0.0	0.0	0.2
0.2	0.0	0.0	0.6	0.2	0.0
0.4	0.0	0.2	0.2	0.2	0.0
0.2	0.4	0.0	0.2	0.6	0.8
P1	P1	P1	P1	P2	P2

Figure 10. An example showing the Uniform Crossover method. The values for each column of the offspring are selected from either one of the parents with a 50% probability.

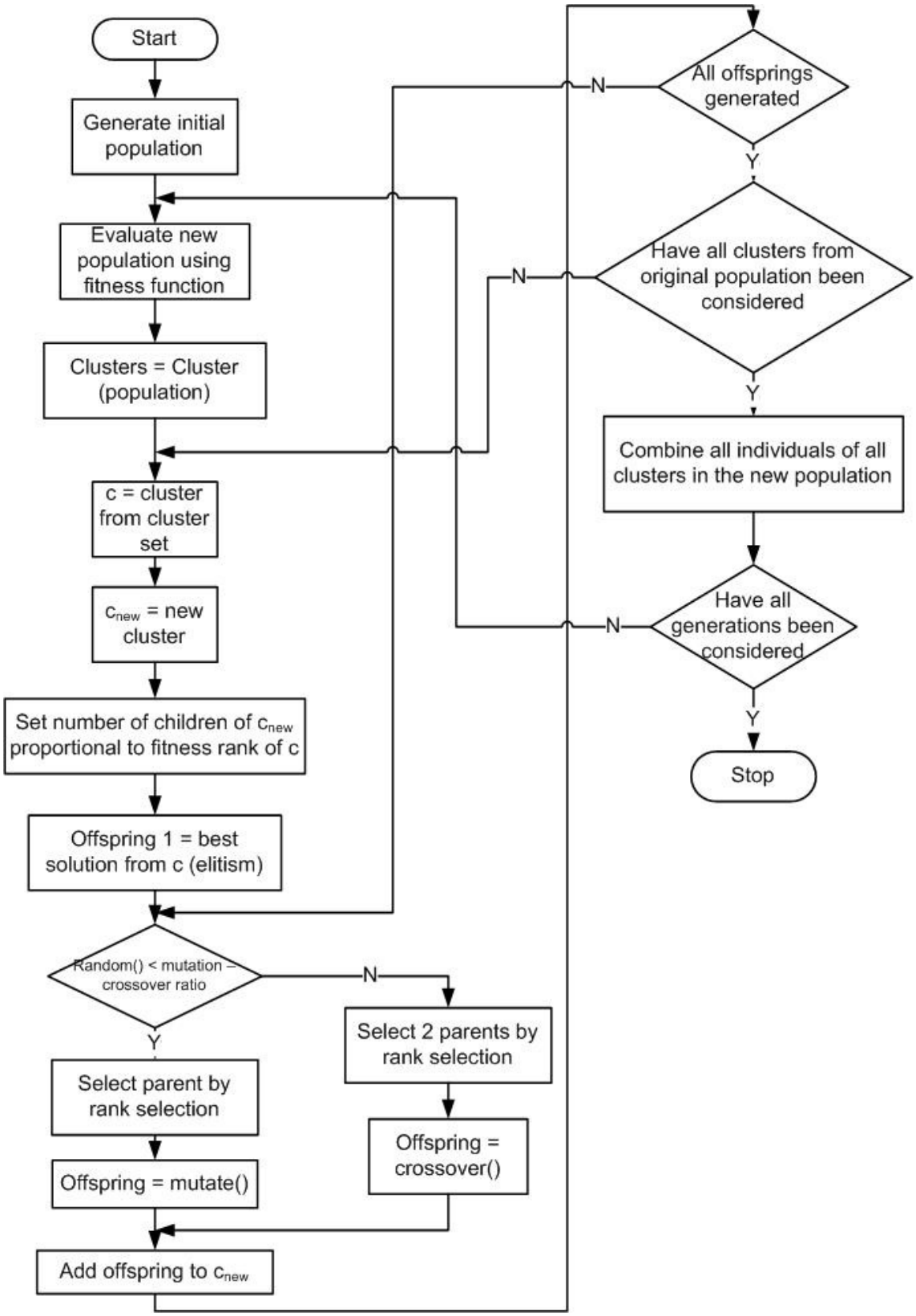

Figure 11. Flowchart of the genetic algorithm being used for motif discovery

Table 1. Various parameters used in the genetic algorithm

Parameter	Value
Length of initial population	8 – 12 bp
Size of initial population	40 PFMs
Mutation to crossover ratio	7 : 3
Mutation1 occurrence rate	Applied 96% of the time when mutation is to occur
Mutation2 occurrence rate	Applied 4% of the time when mutation is to occur
Probability of selecting PFM position during mutation	10% per position. 25% per base after position is been selected
Frequency change permitted for selected base during Mutation1	+/-0.5
Probability of parent selection for each position during crossover	50%
Number of clusters	4

4 Data Sets Used

This section gives details of the various data sets that have been used to check the ability of the algorithm to identify motifs. Four different data sets have been used, of which two are synthetic data sets in which known motifs have been introduced. In the first set, only one motif is embedded. In the second set, multiple motifs have been embedded. The remaining two data sets are a muscle-specific data and a liver-specific data. These data sets were selected because three of the four data-sets have been previously used for researching the motif-discovery issue, and are readily available. The fourth data set (synthetic data set with embedded multiple motifs) was created by inserting motifs at various locations, and checking if the algorithm is able to detect the motifs.

1. Synthetically generated test data with embedded single motifs

   This is a synthetic DNA sequence in which a known motif has been embedded randomly in the input sequences belonging to this set. This data set is used as the simplest case to check if the algorithm can successfully identify the embedded motifs.

   The foreground sequences have been taken from the Eukaryotic Promoter Database (EPD). The motifs are then randomly inserted into more than half of the selected sequences. Two different types of motifs (HFH-1 and HLF) as shown in Table 2 are used in two different data sets. The set of background sequences is comprised of randomly selected sequences from EPD. For this data set, it is ensured that the

number of sequences in both foreground set and background set is the same. Also, the length of the sequences is kept same.

Table 2. Different types of motifs embedded in the synthetic test data

Motif Name	Motif Size	PFM
HFH-1	11	$\begin{bmatrix} 0.2 & 0.7 & 0.3 & 0.2 & 0.0 & 0.0 & 0.0 & 0.0 & 0.9 & 0.0 & 0.3 \\ 0.2 & 0.0 & 0.1 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.1 & 0.0 \\ 0.4 & 0.0 & 0.6 & 0.8 & 0.0 & 1.0 & 1.0 & 1.0 & 0.0 & 0.7 & 0.5 \\ 0.2 & 0.3 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.1 & 0.2 & 0.2 \end{bmatrix}$
HLF	12	$\begin{bmatrix} 0.0 & 0.3 & 0.1 & 0.0 & 0.7 & 0.0 & 0.4 & 0.0 & 0.6 & 0.8 & 0.1 & 0.3 \\ 0.2 & 0.0 & 0.0 & 0.0 & 0.1 & 0.6 & 0.0 & 0.5 & 0.3 & 0.0 & 0.2 & 0.3 \\ 0.3 & 0.0 & 0.9 & 0.8 & 0.1 & 0.2 & 0.0 & 0.5 & 0.0 & 0.2 & 0.6 & 0.3 \\ 0.5 & 0.7 & 0.0 & 0.2 & 0.1 & 0.2 & 0.6 & 0.0 & 0.1 & 0.0 & 0.1 & 0.1 \end{bmatrix}$

** The PFMs for the HFH-1 and HLF motifs have been obtained from the JASPAR database at http://jaspar.cgb.ki.se/*

The length of the foreground and background sequences having the HFH-1 motif embedded in it is 800 bp. The length of the sequences where HLF motif was embedded is 600 bp.

This data set has been used previously for finding motifs by Lones and Tyrrell [13]. It is readily available at http://www.sanger.ac.uk/Software/analysis/nmica.

2. Synthetically generated test data with embedded multiple motifs

This data set is similar to the one above. The foreground sequences obtained are promoter sequences taken from the EPD database. Instances of the two known motifs shown in Table 2 are randomly inserted in a common data set at various locations. This is done to check whether the algorithm is capable of identifying multiple motifs in the same sequence. The HFH-1 and HLF motifs are used for inserting in the data set. The lengths of the sequences in both foreground data set as well as background data set are kept same.

The sequences in both foreground and background data sets have a length of 1000 bp.

3. Liver specific data set

   Both data sets above were created by inserting motifs into sequences. The liver specific data set is used to check how well the algorithm performs when the input data is real biological data. This data set is made up of 91 vertebrate promoter sequences corresponding to genes expressed in the liver. These have also been obtained from the EPD database. The background data set consists of 200, non-liver specific sequences.

   The lengths of sequences for both foreground and background sets are 600bp.

4. Muscle specific data set

   Similar to the liver specific data set, this data set deals with real biological data This data set is made up of 28 vertebrate promoter sequences which were obtained from the EPD database, and are representative of genes that are expressed in muscles. The background data set is also obtained from the EPD database, and is comprised of 200 sequences which are not from the muscle region.

   The lengths of the sequences in both foreground and background data sets are 600 bp.

   Both the above data sets (muscle-specific and liver-specific) have been used by Smith, Sumazin and Zhang [19] when researching the motif discovery issue. They are readily available at http://www.pnas.org/cgi/content/full/0406123102/DC1.

5 Results

For all the data sets, a successful run of the algorithm implies that the evolved final solution must be similar to the consensus sequence. The solution must also have a length greater than 50% of the consensus sequence [13].

1. Synthetically generated test data with embedded single motifs

   a. Test data with embedded HFH-1 motif

      For this data set, runs of the algorithm were made up of 50 and 100 generations. 10 runs were executed for the 50 generations case. 15 runs were executed for the 100 generations case. The population size considered for this experiment was 40. The motifs generated had sizes ranging from 8 – 12 bp. Table 3 summarizes the findings for these experiments.

Table 3. Findings for test runs on HFH-1 embedded synthetic test data

Motif	Sequence Length	Number of generations	Number of runs	Input data set size (sequences)	Background set size (sequences)	Percentage success of the runs
HFH-1	800	50	10	100	100	5/10 = 50%
HFH-1	800	100	15	100	100	13/15 = 87%

A PFM example of the evolved solution for the 50 generations case is:

$$\begin{bmatrix} 0.3 & 0.1 & 0.4 & 0.0 & 0.7 & 0.0 & 0.6 & 0.0 & 0.1 \\ 0.1 & 0.1 & 0.6 & 0.0 & 0.3 & 0.0 & 0.1 & 0.8 & 0.3 \\ 0.2 & 0.4 & 0.0 & 0.3 & 0.0 & 0.0 & 0.3 & 0.1 & 0.4 \\ 0.4 & 0.4 & 0.0 & 0.7 & 0.0 & 1.0 & 0.0 & 0.1 & 0.2 \end{bmatrix}$$

A PFM example of the evolved solution for the 100 generations case is:

$$\begin{bmatrix} 0.4 & 0.5 & 0.2 & 0.0 & 0.0 & 0.0 & 0.6 & 0.0 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.0 & 0.0 & 0.0 & 0.2 & 0.2 & 0.0 \\ 0.4 & 0.0 & 0.6 & 0.0 & 1.0 & 1.0 & 0.0 & 0.6 & 0.5 \\ 0.0 & 0.3 & 0.0 & 1.0 & 0.0 & 0.0 & 0.2 & 0.2 & 0.3 \end{bmatrix}$$

In general, it was observed that the fitness of the evolved motif increased as the number of generations increased. The experiment with 50 generations did not yield a very good solution. The experiment with 100 generations yielded a much better solution. The percentage of successful runs more than doubled with the increase in number of generations.

b. Test data with embedded HLF motif

For this data set, 10 runs were executed for 100 generations. The population size considered for this experiment was 40 PFMs. The motifs generated had sizes ranging from 8 – 12 bp. Table 4 summarizes the findings for this experiment.

Table 4. Findings for test runs on HLF embedded synthetic test data

Motif	Sequence Length	Number of generations	Number of runs	Input data set size (sequences)	Background set size (sequences)	Percentage success of the runs
HLF	600	100	10	100	100	8/10 = 80%

A PFM example for one evolved solution is:

$$\begin{bmatrix} 0.3 & 0.4 & 0.0 & 0.4 & 0.0 & 0.4 & 0.1 & 0.5 & 0.7 & 0.1 \\ 0.1 & 0.0 & 0.1 & 0.2 & 0.6 & 0.0 & 0.5 & 0.4 & 0.0 & 0.3 \\ 0.6 & 0.5 & 0.6 & 0.3 & 0.3 & 0.0 & 0.4 & 0.1 & 0.3 & 0.5 \\ 0.0 & 0.1 & 0.3 & 0.1 & 0.1 & 0.6 & 0.0 & 0.0 & 0.0 & 0.1 \end{bmatrix}$$

In this case, the algorithm returned acceptable results for 100 generations. The algorithm was successful 80% of the time.

2. Synthetically generated test data with embedded multiple motifs

For this data set, only 5 runs of the algorithm were tried. Each run was permitted to execute for 100 generations. Both foreground and background sets contain 100 sequences, each sequence having a length of 1000 bp. The generated initial population contained PFMs for motifs having lengths ranging from 8 – 15 bp. Table 5 summarizes the findings for these experiments for the five runs.

Table 5. Findings for test runs on multiple motifs embedded synthetic test data

Run	Motif HFH-1	Motif HLF
1	Identified	Not Identified
2	Identified	Identified
3	Identified	Identified
4	Not Identified	Identified
5	Not Identified	Identified

As can be seen in the table, these results do not point to any specific trend. In two of the five runs, the algorithm was successfully able to identify both motifs. In two other runs, the algorithm successfully identified the HLF motif only. In one run, it was only able to identify the HFH-1 motif. This shows that the algorithm has only had a success rate of 40% in identifying both motifs in a single run. However, as can be seen, every run was able to identify at least a single motif.

3. Liver specific data set

For this data set, 10 runs of the algorithm were executed. Each run comprised of 100 generations. The foreground data set contained 91 sequences of length 600 bp each. The background data set contained 200 sequences of length 600 bp each. The motifs in the initial population generated had sizes varying in length from 8 to 15 bp. Table 6 gives information about the identified motifs.

Table 6. Findings for test runs on liver-specific data set

No.	Factor	PFM for Factor	Number of runs identified in
1.	FOXJ2	$\begin{bmatrix} 0.6 & 0.4 & 0.3 & 0.2 & 0.4 & 0.4 & 0.0 & 0.5 & 1.0 & 0.6 \\ 0.0 & 0.0 & 0.0 & 0.1 & 0.1 & 0.1 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.4 & 0.3 & 0.4 & 0.5 & 0.0 & 0.0 & 0.6 & 0.5 & 0.0 & 0.2 \\ 0.0 & 0.3 & 0.3 & 0.2 & 0.5 & 0.5 & 0.4 & 0.0 & 0.0 & 0.2 \end{bmatrix}$	8
2.	HNF-1	$\begin{bmatrix} 0.5 & 0.0 & 0.0 & 0.4 & 0.2 & 0.2 & 0.2 & 0.5 & 0.3 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.2 & 0.2 & 0.0 & 0.0 & 0.0 & 0.0 & 0.2 \\ 0.0 & 1.0 & 0.8 & 0.4 & 0.6 & 0.6 & 0.6 & 0.5 & 0.7 & 0.8 \\ 0.5 & 0.0 & 0.2 & 0.0 & 0.0 & 0.2 & 0.2 & 0.0 & 0.0 & 0.0 \end{bmatrix}$	3

** The PFMs for the factors above have been generated from motifs identified in the research paper by Smith, Sumazhin and Zhang [19].*

Only two of the runs were able to identify the two motifs together. Six runs were able to identify only the FOXJ2 factor, and one run was able to identify only the HNF-1 factor. Overall, the FOXJ2 factor was found to be more overrepresented in the liver-specific data set.

An example of the evolved motif similar to the FOXJ2 factor is:

$$\begin{bmatrix} 1.0 & 0.7 & 0.2 & 0.0 & 0.3 & 0.0 & 0.4 & 0.5 & 1.0 & 0.8 \\ 0.0 & 0.0 & 0.2 & 0.2 & 0.3 & 0.3 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.5 & 0.3 & 0.4 & 0.2 & 0.6 & 0.5 & 0.0 & 0.0 \\ 0.0 & 0.3 & 0.1 & 0.5 & 0.0 & 0.5 & 0.0 & 0.0 & 0.0 & 0.2 \end{bmatrix}$$

4. Muscle specific data set

For this data set, 8 runs of the algorithm were executed. Each run was allowed to evolve up to 100 generations. The foreground data set consists of 28 vertebrate muscle specific promoter sequences, of length 600 bp each. The background data set consists of 200 sequences having length 600 bp each. The motifs in the initial population have sizes ranging from 8 to 15 bp. Table 7 gives information about the motifs that have been identified.

Table 7. Findings for test runs on muscle-specific data set

No.	Factor	PFM for Family	Number of runs identified in
1.	MyoD	$\begin{bmatrix} 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.1 & 0.0 \\ 1.0 & 0.0 & 0.6 & 1.0 & 0.0 & 0.0 & 0.4 & 0.6 \\ 0.0 & 0.0 & 0.2 & 0.0 & 1.0 & 0.0 & 0.4 & 0.4 \\ 0.0 & 0.0 & 0.2 & 0.0 & 0.0 & 1.0 & 0.1 & 0.2 \end{bmatrix}$	7

** The PFM for the factor above have been generated from motifs identified in the research paper by Smith, Sumazhin and Zhang [19].*

Of the 8 runs, seven were able to successfully identify a motif similar to the MyoD family. The algorithm was unable to repetitively identify any other motifs, although in one run it identified a motif similar to the Sp1 factor, and in another run, a motif similar to the MEF2 factor was identified.

An example of an evolved motif similar to the MyoD factor is:

$$\begin{bmatrix} 0.4 & 0.4 & 0.0 & 0.0 & 0.8 & 0.0 & 0.0 & 0.2 \\ 0.5 & 0.2 & 0.0 & 0.7 & 0.2 & 0.0 & 0.2 & 0.2 \\ 0.1 & 0.4 & 0.0 & 0.0 & 0.0 & 0.0 & 0.3 & 0.3 \\ 0.0 & 0.0 & 1.0 & 0.3 & 0.0 & 1.0 & 0.5 & 0.3 \end{bmatrix}$$

6 Evaluation

From the results it can be seen that the algorithm is capable of discovering motifs in promoter sequences. In case of single motifs, the algorithm performs reasonably well. In case of sequences having multiple motifs, the algorithm is most successful in identifying the most strongly conserved motif.

Most of the data sets chosen for the project were selected such that they were readily available and had been used earlier for other approaches of motif discovery. Only the data set for synthetic test data with multiple motifs was not available, and had to be generated by inserting the motifs in the promoter sequences.

<u>Synthetic test data with single motif</u>

The results for this data set compare well with the other work done on this data set so far (Lones and Tyrrell [13]). The algorithm performed well for this data set. It was successfully able to identify both HFH-1 and HLF motifs. HFH-1 motif has been recorded as having high information content as compared to HLF. This could be one

reason for the experiment with HFH-1 (87% success rate) being more successful than HLF (80% success rate).

Synthetic test data with multiple motifs

There hasn't been much work done in the area of identifying multiple motifs. Most of the research in this area is directed towards finding the most strongly conserved motifs, and masking these motifs in future runs in order to identify any other motifs present. In all such approaches, multiple executions of the program are necessary, with the discovered motifs being masked every time. The approach of clustering the search space in order to be able to identify multiple motifs within a single run has only been recently proposed [13].

The algorithm was able to identify both multiple motifs in the data sequences in only 2 runs out of 5. This corresponds to a 40% success rate. In the remaining runs, the algorithm was able to identify either one of the two motifs. One of the possible reasons for the algorithm's inability to consistently identify all the multiple motifs present in the input sequences could be the presence of large number of parameters, and the combination of values being used for the various parameters. In most cases (for both single as well as multiple motifs), the initial population size was maintained at 40 motifs. Also, the program was allowed to execute up to 100 generations. The entire search space was being clustered into 4 clusters. It is very likely that the success rate of the algorithm for multiple motifs will be much better if these parameter settings are adjusted better. This will give the algorithm a bigger space to search from. More generations will provide more scope for evolving better solutions. Increasing the number of clusters might also enable better classification of the population for identification of multiple motifs. However, this will require a lot more experimental work with the various combinations of the parameter values to determine the ideal settings for the parameters.

The effectiveness could also be improved by experimenting with different mating methods in the genetic algorithm. Currently the algorithm uses four different methods (elitism, two mutations and one crossover), where mutation and crossover are performed with a ratio of 7:3. Although there are no fixed rules for identifying which methods can prove most useful, experimenting with different techniques might give better results.

The clustering technique chosen in this project is the most basic and simplest methods available, and is faster than most other clustering methods present. It has also been found to be very reliable for clustering a large data set. Using any other clustering method will most likely not affect the effectiveness of this motif-discovery algorithm.

Liver specific test data and Muscle specific test data

The liver specific data set and the muscle specific data set were used to verify how well the algorithm could process real biological data.

Both these data sets have been used previously by Smith, Sumazhin and Zhang [19] for performing motif discovery using an enumerative algorithm that exhaustively searches the whole space of matrices, based on overrepresentation. It was found in the study that the liver-specific data set consisted of three different motifs, of which one was statistically most over-represented (FOXJ2). The muscle specific data set was found to have five different motifs.

For the liver specific data set, it can be seen from the results that the algorithm is able to successfully identify the most over-represented motif (FOXJ2). It was also able to identify one more motif in three of the runs. This shows the algorithm works well for the most over-represented motif, but faces the same issues as the multiple-motif synthetic data set when attempting to identify multiple motifs.

Similar observation can be made with respect to the muscle-specific data set. With the selected settings, the algorithm was successfully able to identify the most over-represented motif (MyoD). But it does not fare very well for multiple motifs. For this specific data set, however, issues in motif identification were anticipated, since it is known that at least two of the other motifs present are very similar to MyoD [13], [19].

Execution details

The project was implemented in C. This language was chosen for its faster execution times as compared to other programming languages.

For this algorithm, bulk of the execution time is needed for evaluating the fitness of the solution. Thus, the time needed for execution grows linearly with the increase in following factors: length of the sequences in the foreground and background sets, number of sequences in the foreground and background data sets, number of motifs in the population, size of the motifs in the population, and the number of generations. Some examples of the execution times encountered are: the synthetic data with single motif required two hours for each run of 50 generations. On increasing the number of generations to 100, the total execution time increased to about 4 hours. The muscle specific data set (91 foreground sequences and 200 background sequences of length 600 bp each) required six hours for execution.

7 Conclusion

In this project, I aimed to identify and use some method for motif discovery that has not been explored much before. In order to achieve this, an evolutionary approach was selected, mainly because this area of research for motif discovery is very new, with little research done as compared to other available methods. Due to its nature, the evolutionary approach has been shown to have much potential in giving good results for this problem. This approach was then combined with clustering to determine the ability to identify better solutions, as well as multiple motifs in the same data set.

This evolutionary approach clusters the search space to maintain diversity in the solutions. This is needed so that unfit solutions are given a chance to evolve into better solutions, as well as to identify multiple solutions. It then attempts to evolve new and better solutions using mutation and crossover.

A series of experiments were performed across diverse data sets to test whether the algorithm achieved its purpose of identifying the motifs present in the biological sequences. The results show that this approach works very well when identifying a single motif from the set of promoter sequence, and is partially successful in identifying multiple motifs.

Selecting the approach for motif discovery was challenging since there are various different techniques that have been suggested in literature. However, with the problem being NP-Complete, none of the approaches suggested so far have had a 100% success rate. Also, the generic nature of the evolutionary approach, though being extremely well suited to this problem type, did pose some challenges as well. Selecting the specific techniques for implementing the genetic algorithm (selection, mutation, crossover and other parameter settings) needed thorough review and research. This is because none of these methods provide a guarantee of the effectiveness of the solution.

I believe that the inconsistency with respect to multiple motifs identification can most likely be improved by experimenting with the different parameter settings. In the end, the project demonstrates a promising approach for motif discovery in biological sequences, and has scope for further exploration.

8 References

[1] Bailey, T. L., Elkan, C. "Unsupervised Learning of Multiple Motifs in Biopolymers Using Expectation Maximization," Machine Learning, 1995, 21(1-2), 51-80.

[2] Buhler, J., Tompa, M. "Finding Motifs Using Random Projections." Journal of Computational Biology, 2002, 9(2), 225 – 242.

[3] Carvalho, A.M., Freitas, A.T., Oliveira, A.L., Sagot, M.F. "An Efficient Algorithm for the Identification of Structured Motifs in DNA Promoter Sequences." IEEE Transactions on Computational Biology and Bioinformatics, 2006, 3(2), 126 – 140.

[4] Davila, J., Balla, S., Rajasekaran, S. "Fast and Practical Algorithms for Planted (l,d) Motif Search". IEEE/ACM Transactions on Computational Biology and Bioinformatics, Oct – Dec 2007, (4), 544-552.

[5] D'haeseleer, P. "What are DNA Sequence Motifs". Nature Biotechnology, Apr 2006, 24(4), 423-425.

[6] "Genetic Algorithms." (Accessed Nov 1, 2008). http://www.tjhsst.edu/~ai/AI2001/GA.HTM

[7] "Gibbs sampling." Wikipedia, The Free Encyclopedia, (Accessed May 19, 2008). http://en.wikipedia.org/w/index.php?title=Gibbs_sampling&oldid=212387203

[8] Hartigan, J. Clustering Algorithms, John Wiley & Sons, 1975.

[9] Hu, J., Li, B., Kihara, D. "Limitations and Potentials of Current Motif Discovery Algorithms." Nucleic Acids Research, 2005, 33(15), 4899 – 4913.

[10] "Introduction to Genetic Algorithms." (Accessed Nov 1, 2008). http://www.obitko.com/tutorials/genetic-algorithms/introduction.php

[11] Krogh, A. "An Introduction to Hidden Markov Models for Biological Sequences." Computational methods in Molecular Biology, Elsevier, 1998, 45-63. (Accessed Nov 10, 2008). http://www.daimi.au.dk/~asger/Krogh98.pdf

[12] Lones, M.A., Tyrrell. "The Evolutionary Computation Approach To Motif Discovery in Biological Sequences." Proc. Genetic and Evolutionary Computation Conf. (GECCO) Workshop Program, Jun 2005, 1-11.

[13] Lones, M.A., Tyrrell. "Regulatory Motif Discovery Using a Population Clustering Evolutionary Approach." IEEE/ACM Transactions on Computational Biology and Bioinformatics, Jul 2007, 4(3), 403-414.

[14] Paul, T.K., Iba, H. "Identification of Weak Motifs in Multiple Biological Sequences Using Genetic Algorithm." GECCO, 2006, 271 – 278.

[15] Pevzner, P.A., Sze, S. H. "Combinatorial Approaches to Finding Subtle

[16] Signals in DNA Sequences," Proc. Eighth Int'l Conf. Intelligent Systems for

[17] Molecular Biology, 2000, 269-278.

[18] Rajasekaran, S., Balla, S. et al. "High-Performance Exact Algorithms for Motif Search." Journal of Clinical Monitoring and Computing, 2005,19(4), 319 – 328.

[19] Raphael, B., Liu, L.T.,Varghese G. "A Uniform Projection Method for Motif Discovery in DNA Sequences." IEEE Transactions on Computational Biology and Bioinformatics, 2004, 1(2), 91 – 94.

[20] Sinha, S., Tompa, M. "A Statistical Method for Finding Transcription Factor Binding Sites." Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology, 2000, 344–354.

[21] Smith, A. D., Sumazin, P., Zhang, M.Q. "Identifying tissue selective transcription factor binding sites in vertebrate promoters." Proceedings of the National Academy of Sciences, Feb 2005, 102(5), 1560-1565.

[22] Tompa, M., Li, N., et al. "Assessing computational tools for discovery of transcription factor binding sites." Nature Biotechnology, 2005, 23(1), 137 – 144.

[23] Zhao, X., Huang, H., Speed, T.P. "Finding Short DNA Motifs Using Permuted Markov Models." Journal of Computational Biology, 2005, 12(6), 894 – 906.

Appendix A

Dry run for a small problem instance

This section gives a dry run for a small problem instance, to give a better practical idea of all steps in the program.

<u>Input data set</u>
Consider the following input data sets consisting of three sequences each
Input set 1 of length 10(foreground sequences corresponding to promoter elements):
IP Sequence 1: A*GTGC*CGTGA
IP Sequence 2: GAGT*GTGCA*T

The italicized portion indicates the motif

Background set:
IP Sequence 1: GAAACCAGCT
IP Sequence 2: ACATTTAGTA

<u>Initial population generation:</u>
The initial population is considered to comprise of 4 PFMs of sizes 4 and 5, generated randomly, representing the motifs.
The table shows the motifs and their corresponding PWMs

PFM1:	$\begin{bmatrix} 0.0 & 0.1 & 0.2 & 0.1 \\ 0.2 & 0.2 & 0.2 & 0.7 \\ 0.0 & 0.5 & 0.2 & 0.2 \\ 0.8 & 0.2 & 0.4 & 0.0 \end{bmatrix}$	PWM1:	$\begin{bmatrix} -\infty & -0.92 & -0.22 & -0.92 \\ -0.22 & -0.22 & -0.22 & 1.03 \\ -\infty & 0.69 & -0.22 & -0.22 \\ 1.16 & -0.22 & 0.47 & -\infty \end{bmatrix}$
PFM2:	$\begin{bmatrix} 0.5 & 0.3 & 0.4 & 1.0 \\ 0.2 & 0.0 & 0.4 & 0.0 \\ 0.1 & 0.2 & 0.1 & 0.0 \\ 0.1 & 0.5 & 0.1 & 0.0 \end{bmatrix}$	PWM2:	$\begin{bmatrix} 0.69 & 0.18 & 0.47 & 1.39 \\ -0.22 & -\infty & 0.47 & -\infty \\ -0.92 & -0.22 & -0.92 & -\infty \\ -0.92 & 0.69 & -0.92 & -\infty \end{bmatrix}$
PFM3:	$\begin{bmatrix} 0.2 & 0.8 & 0.0 & 0.8 & 0.0 \\ 0.6 & 0.0 & 0.8 & 0.0 & 0.0 \\ 0.2 & 0.0 & 0.0 & 0.0 & 0.8 \\ 0.0 & 0.2 & 0.2 & 0.2 & 0.2 \end{bmatrix}$	PWM3:	$\begin{bmatrix} -0.22 & 1.16 & -\infty & 1.16 & -\infty \\ 0.88 & -\infty & 1.16 & -\infty & -\infty \\ -0.22 & -\infty & -\infty & -\infty & 1.16 \\ -\infty & -0.22 & -0.22 & -0.22 & -0.22 \end{bmatrix}$
PFM4:	$\begin{bmatrix} 0.1 & 0.3 & 0.0 & 0.2 & 0.4 \\ 0.0 & 0.3 & 0.0 & 0.0 & 0.4 \\ 0.5 & 0.4 & 0.3 & 0.8 & 0.1 \\ 0.4 & 0.0 & 0.7 & 0.0 & 0.1 \end{bmatrix}$	PWM4:	$\begin{bmatrix} -0.92 & 0.18 & -\infty & -0.22 & 0.47 \\ -\infty & 0.18 & -\infty & -\infty & 0.47 \\ 0.69 & 0.47 & 0.18 & 1.16 & -\infty \\ 0.47 & -\infty & 1.03 & -\infty & -0.92 \end{bmatrix}$

Fitness calculation

The fitness calculation for PWM1:
IP Sequence 1: PWM Match Score at each offset of the input sequence is calculated

Offset	Base	Score
0	A	-∞
1	G	3.36
2	T	-∞
3	G	-∞
4	C	-0.2
5	C	-∞
6	G	1.41

Best Match Score for IP Sequence 1 = 3.36
Normalized score = 1

IP Sequence 2:

Offset	Base	Score
0	G	0.49
1	A	-∞
2	G	2.10
3	T	-∞
4	G	3.36
5	T	-∞
6	G	0.49

Best Match Score for IP Sequence 2 = 3.36
Normalized score = 1

Mean Best Match Score for the first set = (1 + 1)/2 = 3.36

Background Sequence 1:

Offset	Base	Score
0	G	-0.89
1	A	-∞
2	A	-∞
3	A	-∞
4	C	-∞
5	C	0.36
6	A	-∞

Best Match score for Background sequence 1 = 0.36
Normalized score: 0.10

Background Sequence 2:

Offset	Base	Score
0	A	-∞
1	C	-1.59
2	A	-∞
3	T	-∞
4	T	-∞
5	T	-∞
6	A	-∞

Best Match Score for Background Sequence 2 = -1.59
Normalized score = 0.0

Mean Best Match Score = (0.10 + 0.0)/2 = 0.05

Fitness for PFM1:
Mean best match score of input sequence – Mean best match score of background sequence
= 1.00 – 0.05
= 0.95

Similarly, the fitnesses are calculated to the remaining three motifs. These are:
Fitness for PFM2: -0.21
Fitness for PFM3: 0.28
Fitness for PFM4: -0.17

<u>Clustering</u>

Assumption: Only 2 clusters will be formed for this problem instance

The feature vectors for the PFMs are calculated using Algorithm 2.

The mean fitness value for the PFM set = (0.95 -0.21 + 0.28 – 0.17)/4 = 0.21
The PFM with the closest fitness = PFM3.
Thus, PFM3 is made Leader of cluster 1.

Pass 1
The Euclidean distances of all other PFM to the leader are calculated as described in Algorithm 3.
Euclidean distance of PFM1 from PFM3 = 0.34
Euclidean distance of PFM2 from PFM3 = 0.32

Euclidean distance of PFM4 from PFM3 = 0.33

PFM farthest from PFM3 = PFM1.
Thus, PFM1 is made the leader of the second cluster.

Pass 2
PFM3 and PFM1 are the leaders of the two clusters

Euclidean distance of PFM2 from PFM3 = 0.32
Euclidean distance of PFM2 from PFM1 = 0.27

PFM2 is closer to PFM1
Thus, it is assigned to cluster 2 with leader as PFM1

Euclidean distance of PFM4 from PFM3 = 0.33
Euclidean distance of PFM4 from PFM1 = 0.23

PFM4 is closest to PFM1
Thus, it is assigned to cluster 2 with leader as PFM1.

The two clusters formed are:
Cluster 1: {PFM3}
Cluster 2: {PFM1, PFM2, PFM4}

<u>One generation for genetic algorithm</u>

For Cluster 1:
Only one individual is present (PFM3).

Offspring 1

Mutation for PFM3:
Assumption: Mutation1 is applied to PFM 3 (as described in section 3.2.4). Position 2, row 3 is selected for mutation. The frequency value generated for this location is +0.4. The other base values for the row are randomized. Transformation will be as follows:

Parent

$$\begin{bmatrix} 0.2 & 0.8 & 0.0 & 0.8 & 0.0 \\ 0.6 & 0.0 & 0.8 & 0.0 & 0.0 \\ 0.2 & 0.0 & 0.0 & 0.0 & 0.8 \\ 0.0 & 0.2 & 0.2 & 0.2 & 0.2 \end{bmatrix}$$

->

Offspring (PFM11)

$$\begin{bmatrix} 0.2 & \mathit{0.4} & 0.0 & 0.8 & 0.0 \\ 0.6 & \mathit{0.2} & 0.8 & 0.0 & 0.0 \\ 0.2 & \mathit{0.4} & 0.0 & 0.0 & 0.8 \\ 0.0 & \mathit{0.0} & 0.2 & 0.2 & 0.2 \end{bmatrix}$$

This offspring (PFM11) is added to the new population.

For Cluster 2:

Offspring 1

PFM1 has the highest fitness. By elitism, it will propagate to the next generation.

Offspring 2

Assumption: Crossover is applied to PFM1 and PFM4.
The parents selected for each position are: {(Position 1, PFM4), (Position 2, PFM1), (Position 3, PFM1), (Position 4, PFM4), (Position 5, PFM4)}

Parent1 (PFM1) Parent (PFM4) Offspring (PFM21)

$$\begin{bmatrix} 0.0 & 0.1 & 0.2 & 0.1 \\ 0.2 & 0.2 & 0.2 & 0.7 \\ 0.0 & 0.5 & 0.2 & 0.2 \\ 0.8 & 0.2 & 0.4 & 0.0 \end{bmatrix} \begin{bmatrix} 0.1 & 0.3 & 0.0 & 0.2 & 0.4 \\ 0.0 & 0.3 & 0.0 & 0.0 & 0.4 \\ 0.5 & 0.4 & 0.3 & 0.8 & 0.1 \\ 0.4 & 0.0 & 0.7 & 0.0 & 0.1 \end{bmatrix} \text{->} \begin{bmatrix} 0.1 & 0.1 & 0.2 & 0.2 & 0.4 \\ 0.0 & 0.2 & 0.2 & 0.0 & 0.4 \\ 0.5 & 0.5 & 0.2 & 0.8 & 0.1 \\ 0.4 & 0.2 & 0.4 & 0.0 & 0.1 \end{bmatrix}$$

This offspring (PFM21) is added to the new population

Offspring 3

Assumption: Mutation2 is applied to PFM2 (as described in section 3.2.4)
The new column is added at position 2.

Parent Offspring (PFM22)

$$\begin{bmatrix} 0.5 & 0.3 & 0.4 & 1.0 \\ 0.2 & 0.0 & 0.4 & 0.0 \\ 0.1 & 0.2 & 0.1 & 0.0 \\ 0.1 & 0.5 & 0.1 & 0.0 \end{bmatrix} \text{->} \begin{bmatrix} 0.5 & \mathit{0.2} & 0.3 & 0.4 & 1.0 \\ 0.2 & \mathit{0.4} & 0.0 & 0.4 & 0.0 \\ 0.1 & \mathit{0.0} & 0.2 & 0.1 & 0.0 \\ 0.1 & \mathit{0.4} & 0.5 & 0.1 & 0.0 \end{bmatrix}$$

This offspring (PFM22) is added to the new generation.

Thus, the new generation consists of {PFM11, PFM1, PFM21, PFM22}.

<u>Evolution</u>

The above steps of fitness calculation, clustering, reproduction are repeated on the newly generate populations, until the stopping criterion is reached (number of generations).

The final evolved motif is shown below:

PFM for evolved motif

$$\begin{bmatrix} 0.0 & 0.0 & \boldsymbol{0.1} & 0.0 & \boldsymbol{0.4} \\ 0.0 & 0.0 & 0.0 & 1.0 & \boldsymbol{0.4} \\ 0.0 & 1.0 & 0.0 & 0.0 & \boldsymbol{0.2} \\ 1.0 & 0.0 & \boldsymbol{0.9} & 0.0 & 0.0 \end{bmatrix}$$

PFM for consensus sequence

$$\begin{bmatrix} 0.0 & 0.0 & \boldsymbol{0.0} & 0.0 & \boldsymbol{0.5} \\ 0.0 & 0.0 & 0.0 & 1.0 & \boldsymbol{0.5} \\ 0.0 & 1.0 & 0.0 & 0.0 & \boldsymbol{0.0} \\ 1.0 & 0.0 & \boldsymbol{1.0} & 0.0 & 0.0 \end{bmatrix}$$

The italicized bold frequency values indicate the small difference between the evolved motif and the consensus sequence.