U N I V E R S I T Y O F T A R T U
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science

Anton Stalnuhhin

Genetic Algorithm for the Improved Discovery of DNA Regulatory Elements

Bachelor's thesis (4 cp)

Supervisor: Konstantin Tretyakov, B. Sc.

Autor: ..... "....." june 2007
Juhendaja: ..... "....." june 2007

TARTU 2007

Contents

Introduction

Recent decades have been marked with an immense growth of interest in *bioinformatics* and *computational biology* — disciplines dealing with computational analysis of biological data. The reason for that is simple: methods of contemporary molecular biology, such as genome sequencing, gene expression experiments and protein-protein interaction studies, are capable of producing *enormous* amounts of indirect measurements related to many different processes taking place in the cell. However, the data produced in these experiments does not explicitly contain the information of interest: it has to be extracted using sophisticated data analysis techniques. For example, just knowing the DNA sequence for a gene does not yet tell anything about the function of the gene. The sequence has to be analyzed for all kinds of regulatory elements and these, in turn, have to be related to other genes and proteins. This would allow to construct *regulatory networks* of genes which we might then attempt to connect to actual processes and functions in the cells. In short, analysis of biological data is a very complicated task with lots of intermediate steps.

In this work we consider one of these steps, namely the detection of *regulatory elements* in the promoters of the genes, also known as *motifs*. These motifs are short DNA sequences (4-15bp), that are commonly encountered throughout the genome and usually are acting as *binding sites* for certain regulatory molecules, *transcription factors*. Detection of these *transcription factor binding sites (TFBS)* is of utmost importance for biological research and has been a hot research topic for quite a long time already.

A lot of methods have been derived for the detection of TFBS [LW02, THC+06, Vil02, BE94, BWML06, WikiMEME, HETC00]. One of the most common approaches is to search the genome for a number of *overrepresented* patterns and then convert them to a suitable motif representation, such as *position weight matrices (PWM)* or *hidden markov models (HMM)*. In our work we examine a yet another approach to finding high-quality PWM representations for TFBS that aims to *improve* PWMs obtained using other methods with respect to a chosen *goodness measure* with the help of a *genetic algorithm*. This can result in motifs that are better suited for further analysis than the original ones.

This work very closely matches that of [LLB07], if not to say, repeats it. It

should therefore be noted that the original idea and the results of this thesis were obtained independently of that publication. Yet naturally, as the article [LLB07] was published before this thesis was submitted we used the chance to compare our results with the results obtained there [LLB07].

The text of the thesis is split into 4 chapters: at first we present the reader with some important bits of biology required for further understanding. Then we proceed to the description of motif models and position weight matrices in particular. In chapter 3 we describe the idea of a genetic algorithm and in chapter 4 we present the results of practical experiments.

Chapter 1

Biological Background

> Biology is the study of complicated things that give the appearance of having been designed for a purpose
>
> Richard Dawkins, "The Blind Watchmaker" (1986)

1.1 DNA and RNA

Cells are the "building blocks" of life. All living organisms consist of cells. Each cell is a complex system consisting of many different components. Most of the functions in the cell are performed by *proteins* — very large polymeric molecules that can build up tissues, accept and transfer signals or catalyze important reactions. Nowadays it is known that each protein is a chain of aminoacids and the exact sequence of aminoacids for each protein is encoded in the molecules of DNA.

The DNA *(deoxyribonucleic acid)* is a polymeric double-stranded molecule that carries the genetically inheritable information in a cell. Each DNA strand is a chain of monomers, called *nucleotides*, with a backbone made of sugars and phosphate groups. There are four types of nucleotides: *adenine*, *cytosine*, *guanine* and *thymine* (abbreviated A, C, G and T, respectively). Genetic information is encoded by the sequence of nucleotides along the length of the DNA strands. This information contains protein-coding regions (*genes*), areas that regulate gene expression (*promoters, enhancers*) and areas that either have no function, or have no *known* function (*"junk" DNA*).

The RNA *(ribonucleic acid)* is a molecule very similar to DNA. RNA uses the nucleotide *uracil* (U) instead of DNA's *thymine*, the sugar molecule ribose rather than deoxyribose and it has only one strand in contrast to DNA's two. It makes the RNA much more flexible and better suitable for quick transfer of information. The structure differences between DNA and RNA are shown in figure (1.1). Only

one major type of DNA molecule actively participates in the life processes of the cell yet there are several distinct kinds of RNA: the most important of them are *messenger RNA (mRNA)*, *ribosomal RNA (rRNA)* and *transfer RNA (tRNA)*.

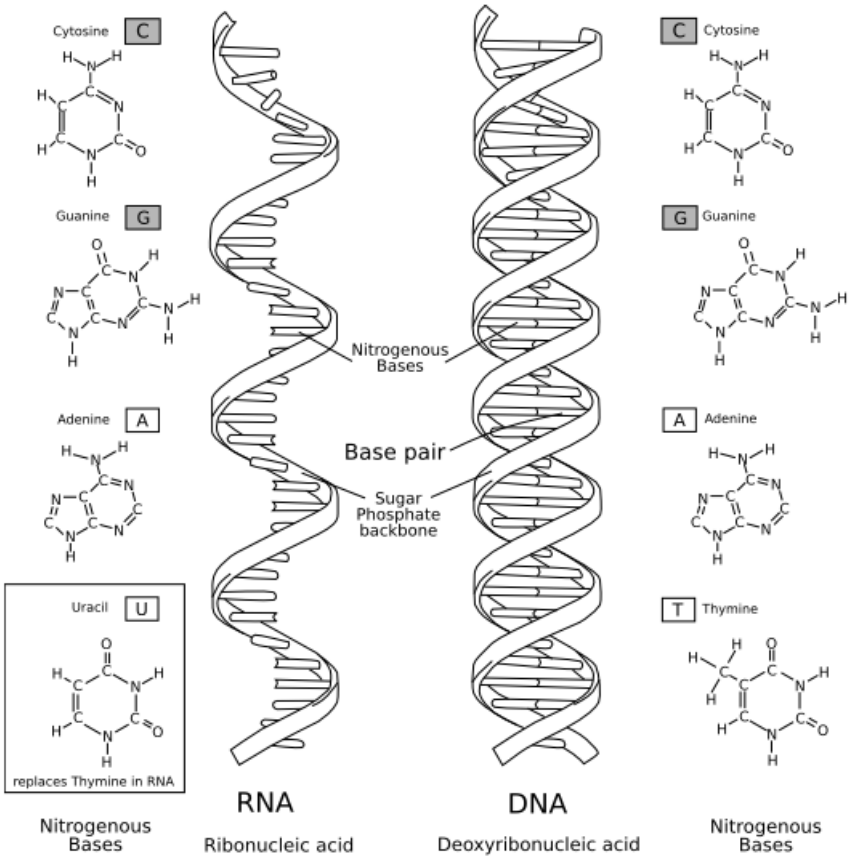
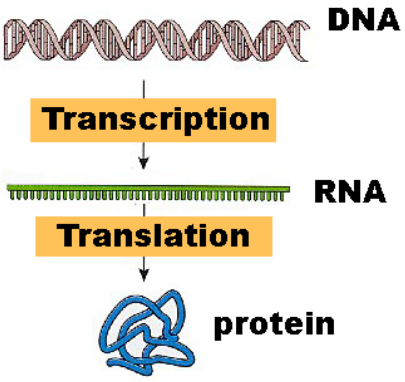


Figure 1.1: Structure of the DNA and the RNA [WikiRNA]

1.2 Transcription and Translation

A *gene* is a region of the DNA encoding the sequence of aminoacids of a certain protein. Although one gene may encode more than one protein, here we regard only the simpler case of "one gene — one protein". The production of a protein from a gene relies on two important processes: *transcription* and *translation*.



Transcription is the process of copying a piece of DNA into a smaller and much more mobile mRNA molecule. This process is divided into four steps:

1. *Unwinding.* The DNA double helix unwinds. One side contains the *coding strand* or the gene of interest, and the other is then non-coding strand.

2. *Base pairing* and *elongation.* Free RNA nucleotides must pair with the coding strand of unwound part of the DNA. The pairing follows the complementarity principle: A always pairs with T, U with A, C with G and G with C. During elongation paired nucleotides are joined together one by one in a single-stranded RNA.

3. *Separation.* The new mRNA molecule separates from the DNA template and the DNA helix reforms itself.

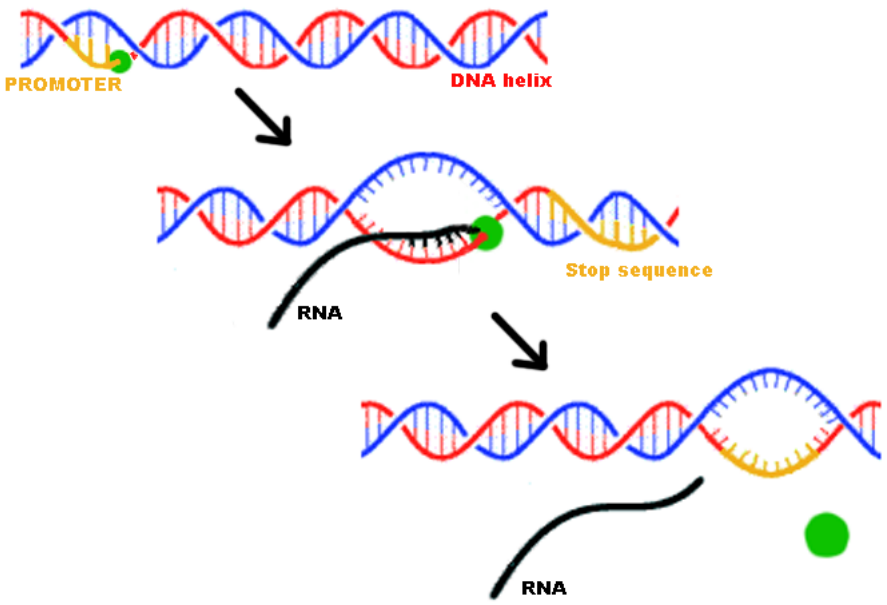


Figure 1.2: Transcription

After some further processing the obtained mRNA molecule is used for protein synthesis — (*translation*). During translation the sequence of nucleotides of the RNA is read by triples, each triple corresponding to a certain aminoacid. A special molecule called the *ribosome* uses the mRNA as a template to connect the appropriate aminoacids in the correct order. After the end of translation the complete aminoacid chain is detached and folded into a ready-to-use protein.

However, it is clearly not necessary for the cell to produce all the proteins at the same time: sometimes certain proteins are needed, at other times — not. Therefore the production of proteins is regulated. This process is called *gene expression regulation*. Gene regulation gives the cell control over structure and function, and is the basis for cellular differentiation, morphogenesis and the versatility and adaptability of any organism. Gene expression can be controlled at many different stages: transcription, translation, post-translational modifications, etc. Here we are only interested in the control of gene expression via the regulation of transcription. Transcription is mainly regulated by *transcription factors* (TF) — special proteins that bind to certain sites on the DNA and thus induce or suppress transcription. The *transcription factor binding sites (TFBS)*, also known as *motifs*, are short sequences of DNA of length about 4–15 bases and are the main focus of this work.

1.3 Computational Detection of TFBS

As it should be clear from the above, detection of transcription factor binding sites in the promoters of the genes is of great interest. This detection can often be done *in vitro* using sophisticated methods of contemporary biology, however biological methods are usually quite expensive, hence a lot of hope is put to the computational techniques.

There are two properties of TFBS we can exploit to detect them automatically. First of all, functional binding sites tend to stay *conserved* through evolution. Hence they can be detected by analysing the promoter sequences of *homologous* (similar) genes of closely related organisms [KPE+03, MWG+06]. Secondly, the binding site for a given transcription factor will tend to be present in the promoter of a number of functionally related genes, therefore it may often be detected by analysing a set of genes with similar expression for overrepresented elements [Vil02]. In this work we consider the latter setting and base ourselves on the results of [Vil02], which our method shall further improve.

Chapter 2

Position Weight Matrices for Motif Representation

2.1 Motif representation

A motif is a short region of the DNA, recognized and bound by a transcription factor. A given TF will often successfully bind to a number of different yet similar short sequences, and therefore a single string is often an insufficient means of representation, many different ways of representing motifs exist. Here are some of them:

- The simplest way to represent a motif is just as a **fixed sequence of nucleotides**. However, as already noted, the binding process is stochastic: a transcription factor might bind both AAAT and AATA with the same strength but not ATAA. Instead of using just the nucleotides in the string, wildcard characters representing several possibilities can be used. For example, in a motif ATWSG, W would mean A or T and S would mean G or C. There is a common standard for such wildcard characters known as the *IUPAC alphabet* [IUP84].

- A more flexible alternative is to represent the motif as a **regular expression** [Fri06], that would allow optional parts (e.g. expression AT?A denotes a motif ATA or AA) and repetitions (e.g. the TC group in the expression A(TC){2,4}A can be repeated 2, 3, or 4 times in the motif).

- Even more general way to represent a binding site is to explicitly enumerate the **set of all possible sequences** to which a factor will bind, e.g. {ATA, AAA, TAA, AAT, AAG, AAC, CAA}. This kind of representation, however, is usually not practical and has more of a theoretical value.

- There are **scoring models** for TFBS representaion. These models use a function that returns a *score* for a given sequence to be a binding site. The most common are *probabilistic scoring models*. These models model either the *probability* or *likelihood* for a given sequence to be a binding site. The two most common examples are *HMM (Hidden Markov Models)* [Mou04] and *PWM (Position Weight Matrices)* [WK03].

In our work we use position weight matrices for motif representation.

2.2 Position Weight Matrices

PWM is a matrix representation of a probabilistic model. When using a PWM, we are modeling the *likelihood* for a given site s to be a *binding site*: $P(s|\text{BS})$ [1]. We assume that the transcription factor binds to each nucleotide in the site independently, in a sense, and therefore the following *position independence* holds: if $s = s_1 s_2 s_3 ... s_n$ is a binding site, then the probability $P(s|\text{BS})$ can be expressed as

$$P(s|\text{BS}) = P(s_1 s_2 s_3 ... s_n|\text{BS}) = P_1(s_1|\text{BS}) P_2(s_2|\text{BS}) P_3(s_3|\text{BS}) ... P_n(s_n|\text{BS}),$$

where $P_i(s_i|\text{BS})$ is the probability that the i'th position in the binding site has nucleotide s_i. Now each probability P_i can be easily estimated from a given set of binding sites:

$$P_i(s_i|\text{BS}) \approx \frac{Q_{s_i,i}}{N},$$

where N is the number of sites in the set, and $Q_{s_i,i}$ is the number of times base s_i is present at position i in the whole set. It is common to consider logarithms of probabilities rather than probabilities themselves and thus replace multiplication with addition. Then:

$$\log_2 P(s|\text{BS}) \approx \sum_{i=1}^{n} \log_2 P_i(s_i|\text{BS})$$

It is also common to take *background* into account. The idea is to compare the probability $P(s|\text{BS})$ for a given s to be a binding site with the probability $P(s|\text{DNA})$ that this site originated randomly from "uninteresting" DNA, by considering the ratio $R(s) = \frac{P(s|\text{BS})}{P(s|\text{DNA})}$. If it is greater than 1 then the site is rather

[1] Formally we are of course rather interested in the *probability* $P(\text{BS}|s)$, however, by Bayes' theorem [BAM01], it's proportional to likelihood $P(s|\text{BS})$ and we prefer the latter form for convenience. We do omit some simple irrelevant technicalities.

a binding site, otherwise it is rather some random piece of DNA. By using the same assumption of independence and taking logarithms we can rewrite the above *likelihood ratio* as a sum:

$$\log_2 R(s) = \log_2 \frac{P(s|\text{BS})}{P(s|\text{DNA})} = \sum_{i=1}^{n} \log_2 \frac{P_i(s_i|\text{BS})}{P(s_i|\text{DNA})} \approx \sum_{i=1}^{n} \log_2 \frac{Q_{s_i,i}/N}{P(s_i|\text{DNA})}$$

The background probabilities $P(s_i|\text{DNA})$ can be computed by counting the base frequency in the whole genome of a given organism. For example, in yeast *(saccharomyces cerevisiae)* $P(A|\text{DNA}) = P(T|\text{DNA}) = 0.31$ and $P(C|\text{DNA}) = P(G|\text{DNA}) = 0.19$.

Commonly a small value is added to the counts to avoid situations where $Q_{s_i,i}$ is 0 and the corresponding logarithm illegal. Usually it is presented as addition of one more sequence to the set with the background probability in each position:

$$\log_2 R(s) = \sum_{i=1}^{n} \log_2 \frac{(Q_{s_i,i} + P(s_i|\text{DNA}))/(N+1)}{P(s_i|\text{DNA})}$$

By collecting the values of s_i in a $4 \times n$ matrix, the calculation above can be conveniently represented as a sum of values taken from the columns of this matrix, as it is shown on figure below (2.1)

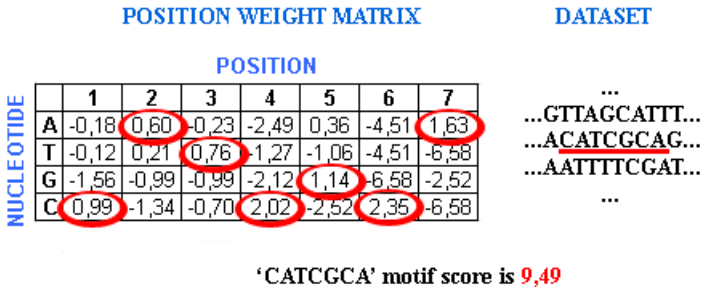


Figure 2.1: Motif scoring example

2.3 Detecting motifs from data

Many motif detection algorithms report PWMs as their results. The most common strategy is to find a viable set of overrepresented sites and combine them into a PWM in the way described above. For example, SPEXS [Vil02] searches a given set of promoters for overrepresented patterns. Suppose some pattern, like 'ATAT.ATATA' has been detected to be overrepresented (i.e. present much more

often in the dataset than it would expected to be in the random or "background" case). By matching this pattern on the data with 1 or 2 mistakes we would get a set of reasonable binding sites, which could then be combined into a PWM. Another popular motif searching tool, MEME [BE94, BWML06, WikiMEME], uses Gibbs sampling [CG92] to detect the most probable PWM for a given set of sequences, assuming each sequence has at least one occurence present somewhere.

PWMs obtained via SPEXS or MEME are usually reasonable and make much sense biologically, however, as they were detected by relying on overrepresentation they may not necessarily be the best for some practical tasks such as functional classification.

Often, we might be interested in using a PWM to actually *detect* putative binding sites and *classify* genes to those that have the site and those that don't. This corresponds to a machine learning-like discriminative problem: we are given a set of "good" genes and a set of "bad" genes and we are interested in finding a PWM that would help *discriminate* the "good" from the "bad". A PWM detected by SPEXS or MEME will probably be quite good at that task, but as those methods did not specifically search for a PWM with good discrimination abilities, they do not report the best possible result.

In this work we shall be focusing on the task of finding PWMs with good discriminative abilities. We shall define a certain measure of discriminative performance and optimize it using a genetic algorithm.

Chapter 3

Genetic Algorithm for PWM Optimization

As it should be clear from the previous chapters, we shall be optimizing a position weight matrix with respect to a number of discriminative performance measures. In this chapter we shall describe the method of optimization we are going to use — genetic algorithm, and specify the measures of performance that we shall be optimizing — the hypergeometric statistic and the ROC area under curve.

3.1 Optimization

The term *optimization*, or *mathematical programming*, refers to the study of problems in which one seeks to minimize or maximize a real-valued function by systematically choosing the values of the arguments. Optimization problem can be represented in the following way:

> Given a set of data X and a function $f : X \to \mathbb{R}$. Find an element k in X such that $f(k) \leq f(x)$ for all x in X. That is called "minimization". Alternatively, find an element k in X such that $f(k) \geq f(x)$ for all x in X. This type of problem is referred to as "maximization". [WikiOptim]

One of the simplest optimization techniques is the following: start at an arbitrary point x and do small steps so that at each step the value of the function increases. This technique is called *hill climbing*. The problem of this algorithm is that it can often end up in a *local maximum*: a suboptimal solution which cannot be improved by making a small step. One way to alleviate the problem is to run the algorithm simultaneously starting from different points (*hill climbing*

with several starting points). An even more sophisticated modification is known as a *genetic algorithm*.

3.2 Genetic Algorithm

A *genetic algorithm (GA)* is a search technique used in computer science to find approximate solutions to optimization and search problems. Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as *inheritance*, *mutation*, *natural selection*, and *recombination* (or *crossover*). [WikiGA]

Two elements are required for any problem before a genetic algorithm can be used to search for a solution. First, there must be a method of representing a solution in a manner that can be manipulated by the algorithm. Traditionally, a solution is represented by a string of bits, numbers or characters. Secondly, there must be a method of measuring the quality of any proposed solution using a *fitness function*.

With these two elements in place, the algorithm proceeds as follows:

1. **Initialization**. Initially many individual solutions are randomly generated to form an *initial population*. The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. Traditionally, the population is generated randomly, covering the entire range of possible solutions (the search space). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found.

2. Repeat until terminating condition.

   (a) **Fitness function**. Evaluate the individual fitnesses of a certain proportion of the population.

   (b) **Selection**. Select pairs of best-ranking individuals to reproduce.

   (c) **Reproduction**. Breed new generation through *crossover* and *mutation*.

Common terminating conditions [WikiGA] are:

- A solution is found that satisfies the optimality criteria.
- Fixed number of generations reached.
- Allocated budget (computation time/money) reached.

- The fitness of the highest ranking solution is reaching or has reached a plateau, so that successive iterations no longer produce better results.
- Manual inspection (watching the process and terminating it manually).
- Combinations of the above.

Selection. During each repetition, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions (measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as this process may be very time-consuming. Most functions are stochastic and designed so that a small proportion of less fit solutions are selected. This helps keep the diversity of the population large, preventing premature convergence on poor solutions. Popular and well-studied selection methods include roulette wheel selection and tournament selection. [WikiGA]

Reproduction: crossover and mutation. The last step in cycle is to generate a second generation population of solutions from those selected through genetic operators: *crossover* (or *recombination*), and *mutation*. For each new solution to be produced, a pair of "parent" solutions is selected for breeding from the pool selected previously. By producing a "child" solution using methods of crossover and mutation, a new solution is created which typically shares many of the characteristics of its "parents". New parents are selected for each child, and the process continues until a new population of solutions of appropriate size is generated. These processes ultimately result in the next generation population that is different from the initial generation. Generally the average fitness will have increased by this procedure for the population, since only the best organisms from the first generation are selected for breeding. [WikiGA]

There are many ways how to perform crossover and mutation. Here we briefly describe some examples how to perform them, if the instances are short strings:

- *Single point crossover* — one crossover point is selected, the part from the beginning of the first parent is copied to the crossover point, the rest is copied from the other parent.

ACCCTGCT+CTGCA**TAT** = **ACCCTTAT**

- *Two point crossover* — two crossover points are selected, the part from the beginning to the crossover point is copied from the first parent, the middle part is copied from the other parent and the rest is copied from the first parent again.

TGAGCTAAA+CTCTATCCA = TGATATCAA

- *Uniform crossover* — information is randomly copied from the first or from the second parent.

TGTAACAACCC+CTCCAACGAGG = TTCCACAGACC

- *Inversion mutation* — selected nucleotides are inverted (for example, $A \to T$, $C \to G$, $G \to C$ and $T \to A$).

TGTAACAACCC = AGTATGTACGG

- *Order changing mutation* — two letters are selected and exchanged.

TGTAACC = CGTAACT

While reproducing a new generation we can use more than one kind of crossover and mutation in any order.

3.3 Genetic Operations for PWMs

As we are searching for an optimal PWM, we have to define the operations of crossover and mutation on PWMs. In this work we have used *count matrices* rather than final PWMs to perform genetic operations. That is, we used the matrices of counts $Q_{s_i,i}$ of nucleotides rather than log-probabilities or log-likelihoods. Naturally, we had to convert the matrices to a log-likelihood form to perform matching. We have used the following operations of reproducing the population:

- *Single, double, triple or four-point crossover* — one, two, three or four crossover points are selected. The parent PWMs are divided into the corresponding number of parts. The new PWM is constructed by alternately taking these parts from parent PWMs.
- *Inversion mutation* — a new PWM is constructed by exchanging rows A and T and rows C and G in the matrix.
- *Random column removal* — a new PWM is a parent PWM with one column removed at a randomly chosen position.
- *Random column addition* — a new PWM is a parent PWM with one random column added at a randomly chosen position. The new column was generated so that the sum of values in this column would equal that of all other columns. This sum was randomly split into 4 parts according to the *multinomial distribution* [Ewe05].
- *Complex operation* — up to 10 randomly chosen operations.

3.4 Fitness Function

In our work we are interested in improving the quality of a motif, hence the fitness function of choice should somehow measure the "goodness" of the motif with respect to the data. There are several reasonable choices of a fitness function. Here we consider two different options: *hypergeometric p-value* and *ROC area under curve*.

3.4.1 Hypergeometric Distribution

Hypergeometric distribution is a discrete probability distribution that describes the number of successes in a sequence of n draws from a finite population without replacement [Kaz04]. It is more convenient to begin with an example.

Think of a basket with two types of balls: blue and red. Let us have N balls, where R are red and $N - R$ are blue ones. Standing next to the basket, we close our eyes and draw n balls randomly. Now we can find the probability P that we draw exactly r red balls and $n - r$ blue ones. If the balls were numbered, we would have $\binom{R}{r}$ options of picking r red balls, $\binom{N-R}{n-r}$ options of picking blue balls and $\binom{N}{n}$ options of picking n balls total. The probability of interest is therefore a ratio of all "good" variants to all the possible ones:

$$P(X = r|R, n, N) = \frac{\binom{R}{r}\binom{N-R}{n-r}}{\binom{N}{n}}$$

We shall also need the probability that a random draw of n balls will result in *at least* r red balls [Sha95]:

$$P(X \geq r|R, n, N) = \sum_{i=r}^{R} \frac{\binom{R}{i}\binom{N-R}{n-i}}{\binom{N}{n}}$$

3.4.2 Hypergeometric p-value

In our case we shall be dealing with a set of sequences divided into 2 parts. The first part contains the promoters of a set of genes that is supposed to be related (e.g. these genes showed similar expression patterns in a microarray experiment). Further on we refer to these as the "good" sequences. The second part contains the promoters of other genes, unrelated to those in the first group, we call them "bad".

For a fixed PWM and a threshold t we can detect the promoters that have at least one match of the matrix with a score greater than t. Let the number of promoters with a match be n and the number of these from the "good" group — r. We say that the PWM is "interesting", if it matches many of the "good" genes and few of the "bad" ones. We estimate it by considering the *p-value* — the probability that if we pick n sequences randomly from the whole dataset, we end up with r or more "good" sequences. It is clear that this probability is expressable using hypergeometric distribution:

$$\text{p-value}(r, R, n, N) = P(X \geq r|R, n, N) = \sum_{i=r}^{R} \frac{\binom{R}{i}\binom{N-R}{n-i}}{\binom{N}{n}},$$

where n and r are described above, N is the number of all sequences and R is the number of "good" ones.

In this work we are optimizing a PWM with no given threshold. Therefore, in order to compute the p-value above, for a given PWM, we search for a threshold resulting in the *minimal* possible hypergeometric probability and use that as the goodness measure[1].

3.4.3 ROC area under curve

Another way of measuring the fitness of a motif is via *ROC area under curve*. We call *true positive* a "good sequence" which was also detected as good (i.e. having a match) by the PWM. A *false positive* is a "bad sequence" which was (probably erroneously) detected to contain a match. With a fixed PWM we can tune the amount of true and false positives by changing the threshold: the higher

[1] Technically speaking, such value is not a proper p-value anymore, but it is still a good performance measure. [Sha95]

the threshold, the less sequences are reported to contain a match, hence the less are the true positive and the false positive count. The plot of *true positive rate* ($\frac{\text{true positive count}}{\text{positive count}}$) versus *false positive rate* ($\frac{\text{false positive count}}{\text{negative count}}$) corresponding to different threshold settings is known as the *ROC (Receiver Operating Characteristic*[2]*) curve*, and it can serve as the indication of the goodness of a classifier.

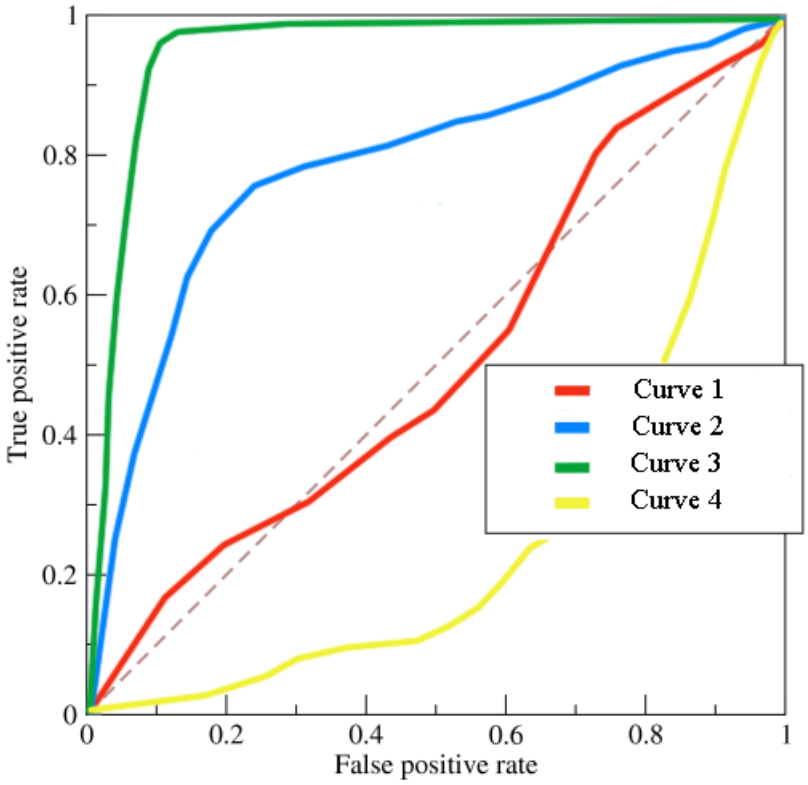


Figure 3.1: The example of ROC curves: *Curve 1* is the line of no discrimination (random guess). *Curve 2* is the better result and *curve 3* is almost perfect. *Curve 4* is the worst one, but it is more interesting, than the random curve (*curve 1*), because *curve 4* has also discriminative results.

If the threshold value is too high, the classifier will produce no positives whatsoever, hence the curve starts at point (0, 0). By lowering the threshold we shall gradually increase the numbers of true and false positives thus making the curve "move" towards the opposite corner (1, 1). If the classifier is really good, then initial lowering of the threshold should bring in much more true positives than false positives: the curve should start up with a steep rise. If the classifier is random, the increase in the rate of true positives will equal that of the false

[2]Receiver Operating Characteristic is a historical name stemming from the popularity of this measure in the analysis of radar signals during the World War II [WikiROC]

positives and the curve will rise at a 45° angle. Finally, if the classifier tends to misclassify, the curve will mostly be located in the lower part of the graph, under the diagonal. The examples of ROC curves are shown on figure above (3.1).

The area under the ROC curve can be shown to represent the probability that, if we pick one "good" sequence and one "bad" sequence, the classifier will tend to score the former higher than the latter. This probability is a very convenient measure of classifier goodness, and thus *ROC area under curve (ROC AUC)* is very well suitable as a fitness function in our case.

Chapter 4

Results

> Argument is conclusive, but it does not remove doubt, so that the mind may rest in the sure knowledge of the truth, unless it finds it by the method of experiment.
>
> Roger Bacon (english philosopher)

We have applied a genetic algorithm on 2 datasets to optimize 4 different initial PWMs. The first dataset is artificial and consists of 40 randomly generated sequences of length 50. 10 of them had the motif $tGATggATgg$ planted somewhere with occasionally introduced errors, and these form the "good" set. The second dataset (*yeast*) consists of 6424 yeast promoters of length up to 603. 98 of them belong to a cluster detected in [Vil02] and these are selected as the "good" part. Three PWMs were detected from this dataset in [Vil02]. We used them as seeds for futher improvement. All results are summarized in the table in **Appendix 1**.

4.1 Artificial PWM

The artifical PWM $tGATggATgg$ is demonstrated as the sequence logo [SS90] in figure (4.1).

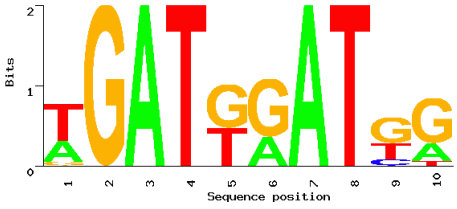


Figure 4.1: Initial PWM

Firstly we attempted to optimize the p-value of this PWM. The genetic algorithm was run for 210 generations with a population size of 50 PWMs and took about 3 hours. As a result the p-value was significantly improved from $2.04 \cdot 10^{-1}$ to $4.68 \cdot 10^{-5}$. The resulting PWM (.....$T.Tg.$) can be seen on the figure below. Note that the obtained PWM also has a considerably higher ROC AUC value than the initial matrix — it went up from 0.44 to 0.84 (see figure (4.4)).

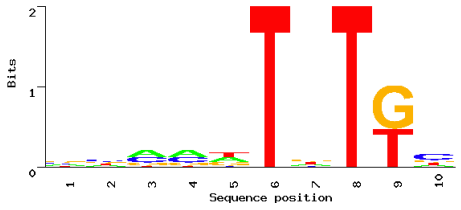


Figure 4.2: Improved (by p-value) PWM

Secondly, we ran the ROC AUC optimization. It lasted 8 hours with 580 generations of 20 PWMs. The obtained PWM (*gc.g.tatcc*) is presented in the figure below.

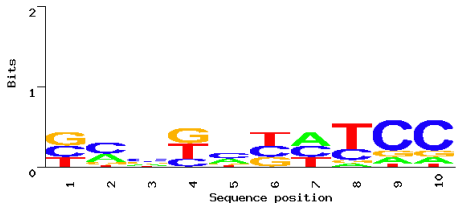


Figure 4.3: Improved (by ROC AUC) PWM

The ROC AUC score got improved from 0.44 to 0.92 (see figure (4.4)). It is also worth noting that the resulting PWM had a p-value even better than the PWM resulting from direct p-value optimization — $4.94 \cdot 10^{-6}$. The reasons for that are the longer duration of optimization in this case, and the close relation between the p-value and the ROC AUC statistics.

As we can see on seqence logos all the PWMs differ quite significantly from each other, hence it might make sense to speak not only of *improvement* of the initial PWM, but rather of the *discovery* of new PWMs.

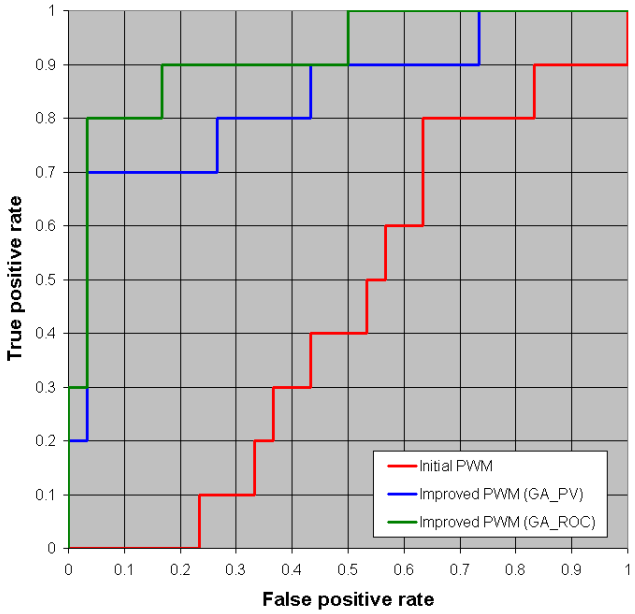
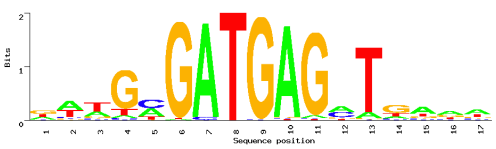


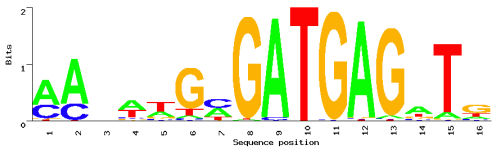
Figure 4.4: ROC AUC for PWMs on artificial dataset

4.2 Yeast PWM #1

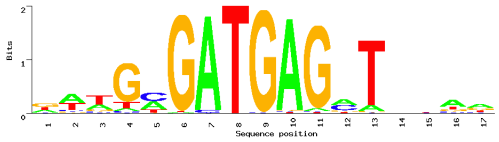
The optimization experiments for this PWM ran longer than all others: 19.5 hours (13 generations with 30 PWMs population) for p-value optimization and about 103 hours (57 generations with 20 PWMs population) for ROC AUC optimization. The results are demonstrated in figure (4.3).



(a) Initial PWM



(b) Improved (by ROC AUC) PWM



(c) Improved (by p-value) PWM

Figure 4.5: PWMs #1 on yeast dataset

The improvement in p-value was notable: from $4.46 \cdot 10^{-45}$ to $2.47 \cdot 10^{-48}$. The p-value did not improve in ROC AUC optimization, yet it remained sufficiently

low — at $2.60 \cdot 10^{-36}$. The ROC AUC increased in both optimizations: from 0.7866 to 0.7842 in p-value optimization and to 0.8057 in ROC AUC optimization test (see figure (4.6)).

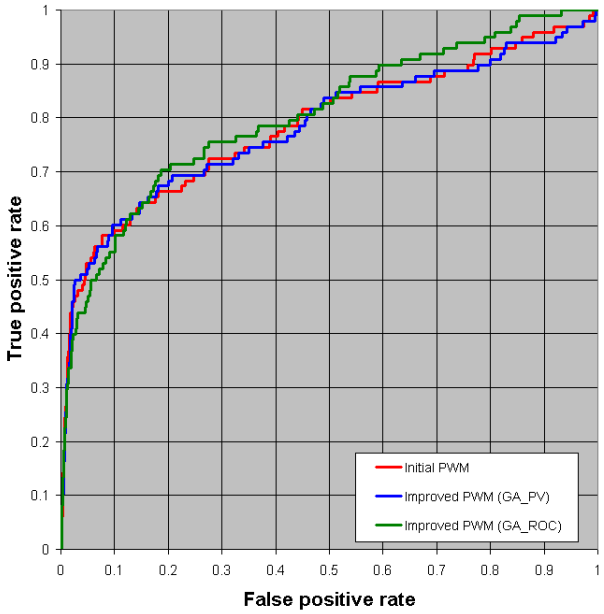
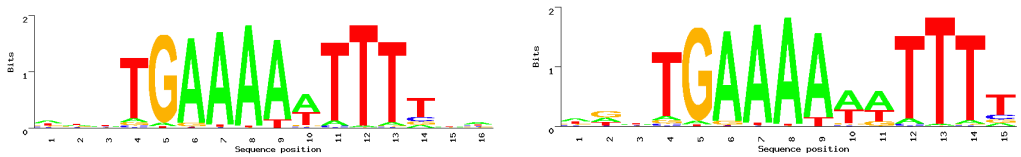


Figure 4.6: ROC AUC for PWMs #1 on yeast dataset

4.3 Yeast PWM #2

During 14.5 hours of p-value optimization (11 generations with 30 PWMs population) no significant improvements of the initial p-value were achieved. Neither was the p-value improved during the ROC AUC optimization (39 hours with 26 generations of 20 PWMs population). The ROC AUC optimization, however, resulted in a minor rise of ROC AUC value from 0.8433 to 0.8561.



(a) Initial PWM (b) Improved (by ROC AUC) PWM

Figure 4.7: PWMs #2 on yeast dataset

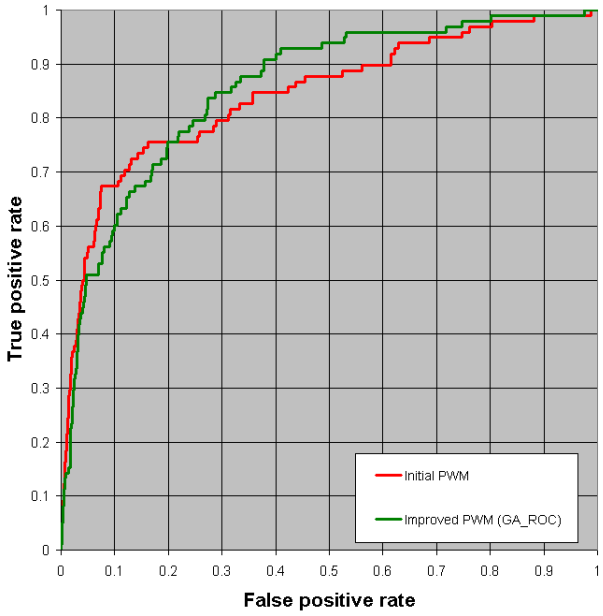
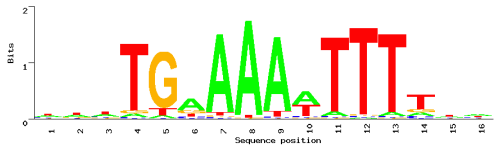


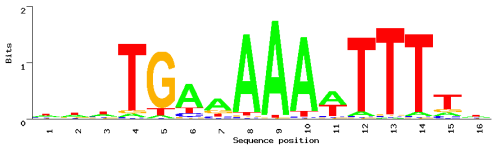
Figure 4.8: ROC AUC for PWMs #2 on yeast dataset

4.4 Yeast PWM #3

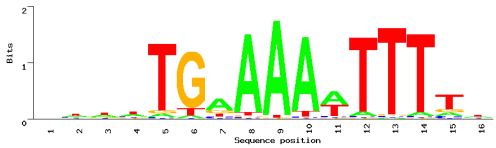
Similarly to the previous matrix, the improvements of both p-value and the ROC AUC score were marginal and insignificant here: in 6 hours (6 generations with 30 PWMs population) of p-value optimization the p-value went from $8.49 \cdot 10^{-44}$ to $7.71 \cdot 10^{-44}$ and ROC AUC went from 0.8438 to 0.8448.



(a) Initial PWM



(b) Improved (by ROC AUC) PWM



(c) Improved (by p-value) PWM

Figure 4.9: PWMs #3 on yeast dataset

The ROC AUC optimization took approximately 7.5 hours (5 generations with population size 25 PWMs). ROC AUC value of derived PWM increased

marginally to 0.8465. The p-value was worsened from $8.49 \cdot 10^{-44}$ to $8.33 \cdot 10^{-35}$. See figure (4.4).

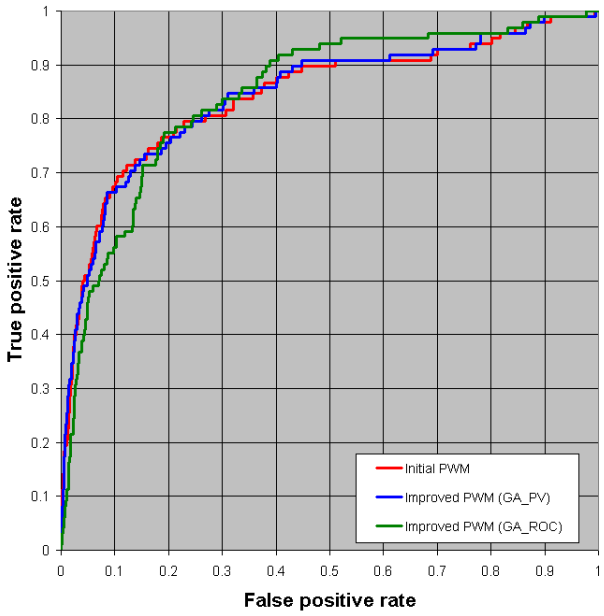


Figure 4.10: ROC AUC for PWMs #3 on yeast dataset

Conclusion

The idea of the work was to examine whether the application of genetic algorithm optimization method can significantly improve PWMs obtained with string-mining methods. We have performed experiments using two different performance measures, and in most cases the measures were improved. In the case of an artificial dataset the result can be seen by eye: the sequence logo of the optimized PWM shows clearly how the initially planted motif was restored.

We find these results very satisfying. In general, genetic algorithm approach turned out to be very convenient and flexible: it does reliably optimize, it is simple to implement, it can be generalized to practically any kind of performance measure, and it can easily be modified for distributed computations. The only negative side is the slowness of the approach. It can take the algorithm hours or days to run if the dataset and the population size are large.

This work provides potential basis for further research and development. It might make sense to search for ways of improving the speed of the algorithm by, say, distributing computations or optimizing the code. Development of a software package that would provide this algorithm for the use of general public is another direction worth looking at.

Genetic Algorithm for the Improved Discovery of DNA Regulatory Elements

Bachelor's thesis

Anton Stalnuhhin

Abstract

Detection of transcription factor binding sites is an important area of contemporary bioinformatics research. Most of the algorithms currently available for that task (e.g. SPEXS or MEME) perform pattern mining on strings, searching for overrepresented or conserved short DNA sequences and reporting the position weight matrices (PWMs), corresponding to the sites found. PWMs thus found can then be used to search for binding sites in other genes or to perform functional classification.

However, the PWMs reported by SPEXS or MEME were not explicitly optimized for discriminative tasks and therefore can be suboptimal. In this thesis we examine a way to optimize these initial PWMs to perform better in gene classification using genetic algorithms.

We used two measures of discriminative performance, hypergeometric p-value and ROC AUC and ran genetic algorithms to optimize them with respect to two datasets: one artificial, and one realistic.

In two experiments out of four the p-value and the ROC AUC score could be significantly improved and we find this result very interesting.

DNA regulatiivsete elementide parendatud otsing kasutades geneetilist algoritmi

Bakalaureusetöö (4 ap)

Anton Stalnuhhin

Resümee

Transkriptsioonifaktori siduvate regioonide tuvastamine on praegu üks olulisematest uurimissuundadest bioinformaatikas. Suurem osa selleks kasutatavatest algoritmidest (nt. SPEXS või MEME) realiseerib mustri kaevandamist andmetes: otsib korduvalt esinevaid voi konserveerunud lühikesi DNA järjestusi ja esitab tulemusi absoluutsete sageduste maatriksitena (PWM). PWM näitab, millise tõenaosusega esineb iga nukleotiid antud positsioonil. Sellisel viisil leitud PWM-d saab kasutada siduvate regioonide otsimiseks teistes geenides või organismide klassifitseerimiseks.

Siiski, SPEXS või MEME abil saadud PWM pole optimiseeritud klassifitseerimise järgi, seega on ta ebaoptimaalne. Antud töös uuritakse ühte PWM parandamise viisi, mis kasutab geneetilist algoritmi. Geneetiline algoritm (GA) on optimiseerimistehnoloogia, mille idee on inspireeritud loodusest: mutatsiooni või ristamise abil produtseeritakse algpopulatsioonist uusi isendeid — sellisel viisil luuakse uus põlvkond, uute isendite jaoks arvutatakse nende sobivuse funktsiooniväärtusi, valitakse hulk parimaid isendeid ja jätkatakse algusest, kus algpopulatsioon on viimane parim hulk. Antud töös defineeritakse geneetilisi operatsioone PWM jaoks.

Proovitud oli kahte erinevat sobivuse arvutuse viisi, et parandada klassifitseerimise võimet: hüpergeomeetriline p-väärtus ja ROC kõvera alla jääv pindala (ROC AUC). Iga mõõtmisega testiti kahte andmestikku: üheks oli tehisandmestik (genereeritud) ja teiseks realistlik (pärm).

Kõik eksperimendid optimiseerisid p-väärtust ja ROC kõvera alla jääva pindala, seega me leiame huvitavaks geneetilise algoritmi kasutamist DNA regulatiivsete elementide otsingus.

References

[BAM01] Geoffrey Berry, P. Armitage, and John Nigel Scott Matthews. *Statistical Methods in Medical Research*. Blackwell Publishing, 2001.

[BE94] Timothy L. Bailey and Charles Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, pages 28–36, 1994.

[BWML06] Timothy L. Bailey, Nadya Williams, Chris Misleh, and Wilfred W. Li. MEME: discovering and analyzing DNA and protein sequence motifs. *Nucleic Acids Res.*, 34:W369–W373, 2006.

[CG92] George Casella and Edward I. George. Explaining the Gibbs Sampler. *The American Statistician*, 46 (3):167–174, 1992.

[Ewe05] Warren John Ewens. *Statistical Methods in Bioinformatics: An Introduction*. Springer, 2005.

[Fri06] Jeffrey E. F. Friedl. *Mastering Regular Expressions*. O'Reilly, 2006.

[HETC00] JD Hughes, PW Estep, S Tavazoie, and GM Church. Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae. *Journal of molecular biology*, 5:1205–1214, 2000.

[IUP84] IUPAC-IUB Joint Commission on Biochemical Nomenclature (JCBN). Nomenclature and symbolism for amino acids and peptides. Recommendations 1983. *Biochem J.*, 219(2):345–373, 1984.

[Kaz04] Leonard J. Kazmier. *Schaum's Outline of Theory and Problems of Business Statistics*. McGraw-Hill Professional, 2004.

[KPE+03] Manolis Kellis, Nick Patterson, Matthew Endrizzi, Bruce Birren, and Eric S. Lander. Sequencing and comparison of yeast species to identify genes and regulatory elements. *Nature*, 423:241–254, 2003.

[LLB07] Leping Li, Yu Liang, and Robert L. Bass. GAPWM: a genetic algorithm method for optimizing a position weight matrix. *Bioinformatics*, 23(10):1188–1194, 2007.

[LW02] Boris Lenhard and Wyeth W. Wasserman. TFBS: Computational framework for transcription factor binding site analysis. *Bioinformatics*, 18(8):1135–1136, 2002.

[Mou04] David W. Mount. *Bioinformatics: Sequence and Genome Analysis*. CSHL Press, 2004.

[MWG+06] Kenzie D MacIsaac, Ting Wang, D Benjamin Gordon, David K Gifford, Gary D Stormo, and Ernest Fraenkelcorresponding. An improved map of conserved regulatory sites for Saccharomyces cerevisiae. *Bioinformatics*, 7:113, 2006.

[Pet93] Pamela Peters. *Biotechnology A Guide to Genetic Engineering*. Wm. C. Brown Publishers, 1993.

[Sha95] J P Shaffer. Multiple Hypothesis Testing. *Annual Review of Psychology*, 46(1):561–584, 1995.

[SS90] T. D. Schneider and R. M. Stephens. Sequence Logos: A New Way to Display Consensus Sequences. *Nucleic Acids Res.*, 18:6097–6100, 1990.

[THC+06] Huai-Kuang Tsai, Grace Tzu-Wei Huang, Meng-Yuan Chou, Henry Horng-Shing Lu, and Wen-Hsiung Li. Method for identifying transcription factor binding sites in yeast. *Bioinformatics*, 22(14):1675–1681, 2006.

[Vil02] Jaak Vilo. *Pattern Discovery from Biosequences*. PhD thesis, 2002.

[WGWZ92] James D. Watson, Michael Gilman, Jan Witkowski, and Mark Zoller. *Recombinant DNA*. 1992.

[WikiDNA] Wikipedia. DNA — Wikipedia, The Free Encyclopedia, 2007. [Online; accessed 1-June-2007]
http://en.wikipedia.org/w/index.php?title=DNA&oldid=134418167.

[WikiGA] Wikipedia. Genetic algorithm — Wikipedia, The Free Encyclopedia, 2007. [Online; accessed 1-June-2007]
http://en.wikipedia.org/w/index.php?title=Genetic_algorithm&oldid=134888373.

[WikiMEME] Wikipedia. Multiple EM for Motif Elicitation — Wikipedia, The Free Encyclopedia, 2007. [Online; accessed 1-June-2007]

http://en.wikipedia.org/w/index.php?title=Multiple_EM_for_Motif_Elicitation&oldid=108103118.

[WikiOptim] Wikipedia. Optimization (mathematics) — Wikipedia, The Free Encyclopedia, 2007. [Online; accessed 1-June-2007]

http://en.wikipedia.org/w/index.php?title=Optimization_%28mathematics%29&oldid=133647046.

[WikiROC] Wikipedia. Receiver operating characteristic — Wikipedia, The Free Encyclopedia, 2007. [Online; accessed 1-June-2007]

http://en.wikipedia.org/w/index.php?title=Receiver_operating_characteristic&oldid=134552700.

[WikiRNA] Wikipedia. RNA — Wikipedia, The Free Encyclopedia, 2007. [Online; accessed 1-June-2007]

http://en.wikipedia.org/w/index.php?title=RNA&oldid=134466453.

[WK03] David D. Womble and Stephen A. Krawetz. *Introduction to Bioinformatics: A Theoretical and Practical Approach.* Humana Press, 2003.

Internet URL-s of the references were still valid on June 1, 2007.

Appendices

Appendix 1. Table of all results and values for initial and recieved PWMs.

Appendix 2. Java implementation of the methods for discovering the PWM using genetic algorithm and the results of the experiment are on the CD.

Appendix 1

All tables have descriptions for three PWMs set in three rows: initial PWM (**INIT**), PWM recieved by improving p-value (**PV**) and PWM recieved by improving ROC AUC (**ROC**). PWM description consists of the following components: **PV** is for p-value, **PVT** is for the optimal score threshold used during p-value calculation, **PVG** is for portion of matches in "good" sequences, **PVB** is for portion of matches in "bad" sequences, **ROC** is for ROC AUC value, **G** is for the number of generations in experiment, **P** is for the number of PWMs in population and **D** is for duration of the experiments (in hours).

PWM on artificial data.

	PV	**PVT**	**PVG**	**PVB**	**ROC**	**G**	**P**	**D**
INIT	$2.04 \cdot 10^{-1}$	-2.09	8/10	19/30	0.4433	-	-	-
PV	$4.68 \cdot 10^{-5}$	3.65	7/10	1/30	0.8400	210	50	3
ROC	$4.94 \cdot 10^{-6}$	2.77	8/10	1/30	0.9167	580	20	8

PWM #1 on yeast data.

	PV	**PVT**	**PVG**	**PVB**	**ROC**	**G**	**P**	**D**
INIT	$4.46 \cdot 10^{-45}$	10.44	43/98	114/6325	0.7866	-	-	-
PV	$2.47 \cdot 10^{-48}$	9.51	49/98	160/6325	0.7842	13	30	19.5
ROC	$2.60 \cdot 10^{-36}$	8.56	42/98	183/6325	0.8057	57	20	103

PWM #2 on yeast data.

	PV	**PVT**	**PVG**	**PVB**	**ROC**	**G**	**P**	**D**
INIT	$2.12 \cdot 10^{-48}$	8.49	66/98	478/6325	0.8433	-	-	-
PV	$2.12 \cdot 10^{-48}$	8.49	66/98	478/6325	0.8433	11	30	14.5
ROC	$7.58 \cdot 10^{-38}$	8.93	50/98	303/6325	0.8561	26	20	39

PWM #3 on yeast data.

	PV	**PVT**	**PVG**	**PVB**	**ROC**	**G**	**P**	**D**
INIT	$8.49 \cdot 10^{-44}$	8.02	64/98	517/6325	0.8438	-	-	-
PV	$7.71 \cdot 10^{-44}$	7.97	65/98	543/6325	0.8448	6	30	6
ROC	$8.33 \cdot 10^{-35}$	6.55	76/98	1217/6325	0.8465	5	25	7.5