A Genetic Algorithm for Motif Finding Based on Statistical Significance

Josep Basha Gutierrez[1], Martin Frith[2], and Kenta Nakai[3]

[1] Department of Medical Bioinformatics,
Graduate School of Frontier Sciences, The University of Tokyo,
5-1-5 Kashiwanoha, Kashiwa-shi, Chiba-ken 277-8561, Japan
yusef@hgc.jp
[2] Computational Biology Research Center,
AIST Tokyo Waterfront Bio-IT Research Building,
2-4-7 Aomi, Koto-ku, Tokyo, 135-0064, Japan
[3] Human Genome Center, The Institute of Medical Science,
The University of Tokyo, 4-6-1 Shirokane-dai,
Minato-ku, Tokyo 108-8639, Japan
knakai@ims.u-tokyo.ac.jp

Abstract. Understanding of transcriptional regulation through the discovery of transcription factor binding sites (TFBS) is a fundamental problem in molecular biology research. Here we propose a new computational method for motif discovery by mixing a genetic algorithm structure with several statistical coefficients. The algorithm was tested with 56 data sets from four different species. The motifs obtained were compared to the known motifs for each one of the data sets, and the accuracy in this prediction compared to 14 other methods both at nucleotide and site level. The results, though did not stand out in detection of false positives, showed a remarkable performance in most of the cases in sensitivity and in overall performance at site level, generally outperforming the other methods in these statistics, and suggesting that the algorithm can be a useful tool to successfully predict motifs in different kinds of sets of DNA sequences.

Keywords: Motif finding, Genetic Algorithm, Transcription Factor Binding Site, Statistical significance.

1 Introduction

Sequence motifs are short nucleic acid patterns that are repeated very often and have some biological significance. Their function is usually to serve as sequence-specific binding sites for proteins such as transcription factors (TF). The discovery of these sequence elements in order to get a better understanding of transcriptional regulation is a fundamental problem in molecular biology research. Traditionally, the most common methods to determine binding sites were DNase footprinting, and gel-shift or reporter construct assays. Currently, however, the use of computational methods to discover motifs by searching for

F. Ortuño and I. Rojas (Eds.): IWBBIO 2015, Part I, LNCS 9043, pp. 438–449, 2015.

overrepresented (and/or conserved) DNA patterns in sets of functionally related genes (such as genes with similar functional annotation or genes with similar expression patterns) considerably facilitates the search. The existence of both computationally and experimentally derived sequence motifs aplenty, as well as the increasing usefulness of these motifs in the definition of genetic regulatory networks and in the decoding of the regulatory program of individual genes, make motif finding a fundamental problem in the post-genomic era.

One of such many strategies for motif discovery relies on the use of Genetic Algorithms (GA).

Genetic Algorithms. A genetic algorithm is a search heuristic that tries to imitate the process of natural selection in order to find exact or approximate solutions to optimization or search problems. The motivation for using genetic algorithms comes from the idea of reducing the number of searches in a high number of large DNA sequences.

The basic structure of a genetic algorithm consists of evolving a population of candidate solutions (individuals) in order to find the best solution or set of solutions possible. This is performed through an iterative process in which the population in each iteration will be considered a generation. In each one of these generations, the fitness (the score given to measure how good the individual is as a solution for the problem) of every individual in the population is evaluated. The fittest individuals are selected from the current population, and a new generation is created by crossover and mutation of these fit individuals. The new generation is then used in the next iteration of the algorithm. The algorithm will normally terminate when either a satisfactory solution has been found or a maximum number of generations has been reached. The main challenge therefore resides in successfully defining the population, and the fitness, crossover and mutation functions.

The most common approach for motif finding using Genetic Algorithms [1] assumes that every input sequence contains an instance of the motif, and relies on the following elements:

- Each individual is represented by a vector $P = \{p_1, p_2, ..., p_N\}$ storing the starting positions for each one of the TFBS instances for the given set of N sequences $S = \{S_1, S_2, ..., S_N\}$. Thus P represents a possible solution set $M = \{m_1, m_2, ..., m_N\}$, where each m_i is an instance with length w from sequence S_i.
- The fitness of each individual is computed using the similarity score of the consensus string produced by an individual, using the PWM (position weight matrix).

$$Fitness(M) = \sum_{i=1}^{w} f_{max}(i) \tag{1}$$

where M is a candidate motif, w is the motif length and $f_{max}(i)$ is the maximum frequency value in column i in the PWM.

Current Situation. As well as the standard GA, most of the existing motif finding methods deal with a set of DNA sequences in which all of the sequences (or at least most of them) are expected to contain at least one instance of each one of the transcription factor binding sites (TFBS) that the method will report, sustaining the search on the statistical enrichment of these TFBS in the set of sequences.

In the last years, new statistical properties of TFBS have been discovered [2]. By the use of this statistical information, this research explores the efficiency of the application of a different sort of genetic algorithm, with fewer restrictions about the input sequences, the presence of instances and the size of the datasets, and able to work with large datasets without consuming a great amount of time. In the method here described, we would like to find TFBS based on their statistical enrichment in any of the input sequences (not necessarily most of them). For that purpose, a new GA-based method was designed, in which the sequences are treated iteratively, creating random subsets of them in a random order and analyzing the statistical overrepresentation in several steps.

2 Methods

The method here proposed mixes a genetic algorithm with probabilistic methods, trying to integrate the advantages of both.

The main characteristics of the method are the following:

- There are no assumptions about the presence of the motifs in the input sequences. Unlike other methods, which assume that every sequence contains at least one instance of the motif, in this method the motifs can be distributed in any way in the given sequences, with the only assumption that they are overrepresented in at least a few of them.
- It is a heuristic algorithm. Thus, it may produce different results each time it is run
- Individual motifs are ungapped. Patterns with variable-length gaps might be predicted split into two or more separate motifs.
- The background set of sequences is generated dynamically throughout the process by shuffling the candidate motifs to analyze against the sequences instead of shuffling the sequences themselves.

2.1 Representation

To represent the candidate motifs, initially there were two possible options: either using a string with the sequence of nucleotides, or using a position in which the instance is located. As Vijayvargiya and Shukla [1] proved in their experiment, the approach with positions is more appropriate for a genetic algorithm (GA). Therefore, that was the approach chosen for our method.

However, in that standard algorithm, as it assumed there was one instance of the motif in each of the input sequences, each individual was represented by a

vector with the beginning positions of each one of those instances. In the method here described, on the other hand, there are no assumptions about the presence of instances in every sequence, so the individuals are represented by a single position value in what we call the *supersequence*.

Supersequence. In order to deal with that, we needed to have a structure in which a single value for that position represents a unique individual for the whole set of sequences. For that purpose, all the input sequences are joined in a single supersequence before starting the algorithm. That way, each individual will be represented by a unique position in the supersequence. This position, at the same time, represents the motif given by the position itself, a fixed motif length and a maximum number of mutations allowed.

It is important to clarify that the supersequence serves only as a means of representation of the motifs during the algorithm process and there is no biological meaning in it. The final solutions will be represented by a position weight matrix, got by clustering all the predicted instances with a high level of similarity.

Subsequences. In order to discard unfit individuals faster to generate more diverse solutions, the supersequence is divided in subsequences of an arbitrary length regardless of the length of each sequence (defined by a parameter that by default has a value of 500 bp).

For each generation of the population, the fitness will be calculated against one of the subsequences. In other words, in each iteration the algorithm will search for overrepresentation of the motifs within the given subsequence. The purpose of creating the subsequences is only to simplify the fitness function and, as well as the supersequence, there is no biological meaning in it.

The order of the original sequences will be shuffled every S generations, being S the total number of subsequences.

Fig. 1 shows how the supersequence and the subsequences are created.

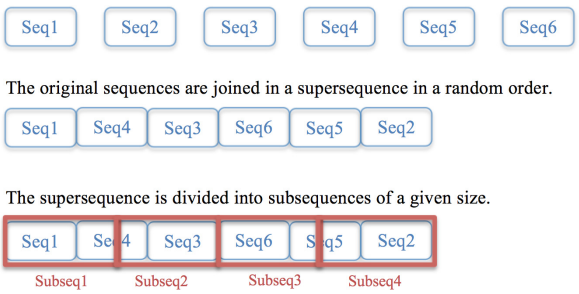

Fig. 1. Creation of the supersequence and the subsequences

2.2 Evaluation and Selection

The algorithm does not have a typical fitness function, but a combination of different ones applied at different moments of the process.

Simple Overrepresentation. The first step to measure the quality of the solutions consists of checking if they are overrepresented in the given subsequence. In order to calculate it, the algorithm follows these steps for each individual of the population:

1. Getting the candidate motif i given by the position P of the individual and the fixed motif length l.
2. Shuffling the candidate motif to get a background motif $b(i)$.
3. For both the candidate motif and the background motif, counting the number of similar words ($S_W(x)$) in the given subsequence (those words with length l that are exactly the same as the motif except for as much as m mismatches, being m the number of mismatches allowed).
4. Storing both values ($S_W(i)$ and $S_W(b(i))$) in vectors that are kept along with the individual.
5. Calculating the difference of similar words between the candidate motif and the background motif ($N(i)$):

$$N(i) = S_W(i) - S_W(b(i)) \tag{2}$$

Candidate Selection (First Fitness Measurement). In order to select the best candidate solutions as survivors and generate new individuals by crossover of these fit candidates, the *Fluffiness Coefficient* ($F_N(i)$), inspired by the"fluffy-tail test" proposed by Abnizova et al. [3], is used for every individual in every generation.

$$F_N(i) = \frac{N(i) - \mu_s}{\delta_s} \tag{3}$$

where μ_s and δ_s are, respectively, the mean and the standard deviation of the Simple Overrepresentation values ($N(i)$) for the set of solutions. The individuals with the lowest Fluffiness value are eliminated from the population.

Solution Selection (Second Fitness Measurement). Once an individual has survived for at least 10 generations, a new fitness test is performed in order to decide if the candidates are final solutions to the problem or not. For that purpose, two different coefficients are used.

– **Thinness Coefficient.** This coefficient is inspired by the "thin-tail test" proposed by Shu and Li [4]

$$T_N(i) = \frac{k_0 - 2\epsilon}{4\epsilon} \text{ where } \epsilon = 2\sqrt{\frac{6}{M}} \text{ and } k(i) = \frac{M(f_Z(i) - \mu_s)^4}{(M-1)\delta_s^4} - 3 \tag{4}$$

M is the total number of individuals in the population, k(i) is the kurtosis of the individual i and $f_Z(i)$ is the number of individuals in the population with the same fitness value $N(i)$ as i. The individuals with a Thinness coefficient above 0.6 are eliminated from the population.

– **Mann-Whitney U Test.** The Mann-Whitney U test (also known as Wilcoxon rank-sum test) is a nonparametric test that, for two populations, measures if one of them tends to have larger values than the other.

$$U_1 = n_1 n_2 + \frac{n_1(n_1+1)}{2} - R_1 \tag{5}$$

where n_1 is the sample size for the sample 1, and R_1 is the sum of ranks in the sample 1.
In our algorithm, the first sample corresponds to the vector with the values stored for the number of similar words for the candidate motif in each generation, and the second sample is formed by the same values for the background motif.
If the probability of both data samples coming from the same population is lower than 0.05, then the motif is considered as a possible final solution.
The final fitness value will be given by the following formula:

$$FV(i) = p_{val}(i) \times w \tag{6}$$

where $p_{val}(i)$ is the Mann-Whitney U test p-value and w is the motif width.

2.3 Genetic Operators

Crossover. The crossover function is a one-point crossover in which a child is generated by the parts of both parents joined in reversed order. The parents are the motifs given by the position of the individuals, and the position of the newborn child will be the position of the most similar word in the supersequence.

Mutation. Mutation will happen randomly, according to a parameter that defines the frequency. It will also be applied to random individuals. The mutated individual will slightly change its position by a random offset between 1 and the motif length

2.4 Post Processing

Filtering and Clustering of Solutions. After running the algorithm for every given motif width, the solutions are filtered and clustered to generate the final solutions, each one of them formed by a combination of instances (preliminary solutions given by the GA) with a high level of similarity. The fitness of the clustered solution will be the maximum of the fitnesses of the motifs that are part of the cluster.

In order to devise the similarity, the algorithm measures the distance between motifs.

The distance between motifs is defined by the sum of the distances between their IUPAC symbols. Each IUPAC symbol represents a subset of the symbols {A, G, C, T}. The distance between two symbols s_1 and s_2 is calculated as follows:

$$d(s_1, s_2) = 1 - 2\frac{|s_1 \cap s_2|}{|s_1| + |s_2|} \tag{7}$$

So the values will be always between 0 and 1, being 0 for identical symbols and 1 for disjoint symbols.

The distance between two motifs x and y with sizes m and n respectively is calculated as follows:

$$D(x, y) = \sum_{i=1}^{min(m,n)} d(x_i, yi) + w_u u \tag{8}$$

where u is the number of unpaired bases ($|m - n|$) and w_u is the weight assigned to them (0.6 by default). All possible shifts aligning the motifs are considered and the final distance will be the minimum.

Two motifs are considered similar (and clustered in the same motif) if:

$$\frac{D(x, y)}{mean(|x|, |y|)} \leq MS \tag{9}$$

where MS is the Maximum Similarity, a parameter that can be adjusted and that, by default, will be 0.5.

3 Results

Assessment. The tool was tested using the assessment provided by the study performed by *Tompa et al.* [5] to compare the accuracy of motif finding methods. This assessment provides a benchmark containing 52 data sets of four different organisms (fly, human, mouse and yeast) and 4 negative controls. The data sets are of three different types: the real promoter sequences in which the sites are contained (Type Real), random promoter sequences from the same genome (Type Generic) and synthetic sequences generated by a Markov chain of order 3 (Type Markov). The assesment compared the efficiency of 14 methods, to which we compared our method as well. The eight statistics that will define the accuracy of each tool are the following ones:

- nSn (Sensitivity, nucleotide level), gives the fraction of known site nucleotides that are predicted:

$$nSn = \frac{nTP}{nTP + nFN} \tag{10}$$

– $nPPV$ (Positive Predicted Value, nucleotide level), gives the fraction of predicted site nucleotides that are known:

$$nPPV = \frac{nTP}{nTP + nFP} \tag{11}$$

– nSp (Specificity):

$$nSp = \frac{nTN}{nTN + nFP} \tag{12}$$

– nPC (Performance Coefficient, nucleotide level) [6]:

$$nPC = \frac{nTP}{nTP + nFN + nFP} \tag{13}$$

– nCC (Correlation Coefficient) [7]:

$$nCC = \frac{nTP \times nTN - nFN \times nFP}{(nTP + nFN)(nTN + nFP)(nTP + nFP)(nTN + nFN)} \tag{14}$$

– sSn (Sensitivity, site level), gives the fraction of known sites that are predicted:

$$sSn = \frac{sTP}{sTP + sFN} \tag{15}$$

– $sPPV$ (Positive Predicted Value, site level), gives the fraction of predicted sites that are known:

$$sPPV = \frac{sTP}{sTP + sFP} \tag{16}$$

– sASP (Average Site Performance) [7]:

$$sASP = \frac{sSn + sPPV}{2} \tag{17}$$

Where TP refers to the number of true positives, FP refers to the number of false positives, TN refers to the number of true negatives, and FN refers to the number of false negatives. The n before each one of these measures refers to *nucleotide level* and the s refers to *site level*.

Tests. Our algorithm was run 3 times for each data set, using different motif lengths, and then all the results combined in the post processing stage.

– Motif width 8, allowing 2 mismatches
– Motif width 10, allowing 3 mismatches
– Motif width 12, allowing 4 mismatches

The parameters with which the algorithm was run for all of the data sets are the following:

- Population size: 200
- Number of generations: 100
- Maximum number of solutions: 100
- Mutation rate: 0.1
- Subsequence size: 500bp
- Maximum Similarity: 0.7

Fig. 2 summarizes the average values of the mentioned statistics got by each one of the 14 methods of the assessment and our own method regardless of the organism and the type of data set. Fig. 3 shows the average values grouped by organisms. The procedure to calculate the average in every case is as follows: The values of nTP, nFP, nFN, nTN, sTP, sFP and sFN of the different data sets are added, and then the given statistic is computed as if the summed values corresponded to a unique large data set.

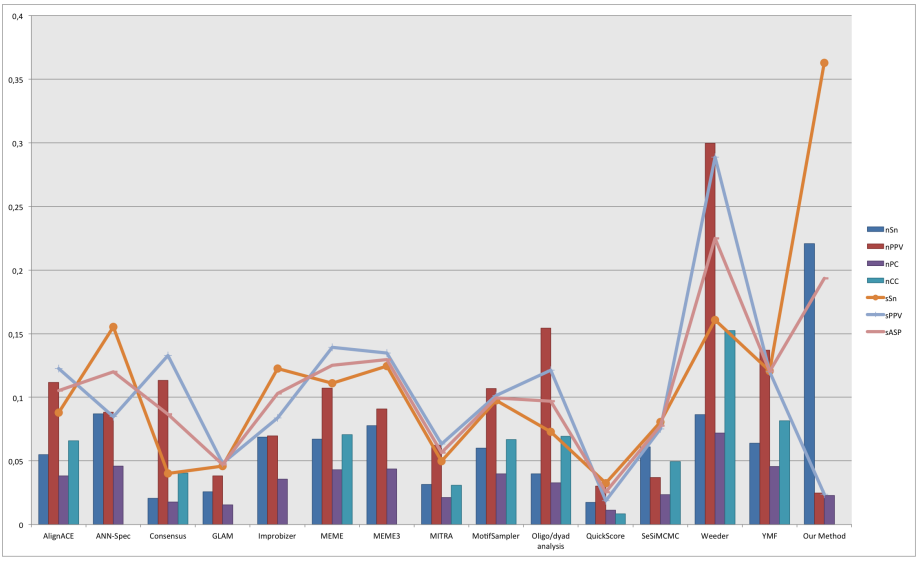

Fig. 2. Average statistical values for all 56 data sets

4 Discussion

First of all, as the authors of the assessment [5] in which our tests are based explain, these statistics should not be taken as an absolute measurement of the quality of the methods. There are many factors that affect the results:

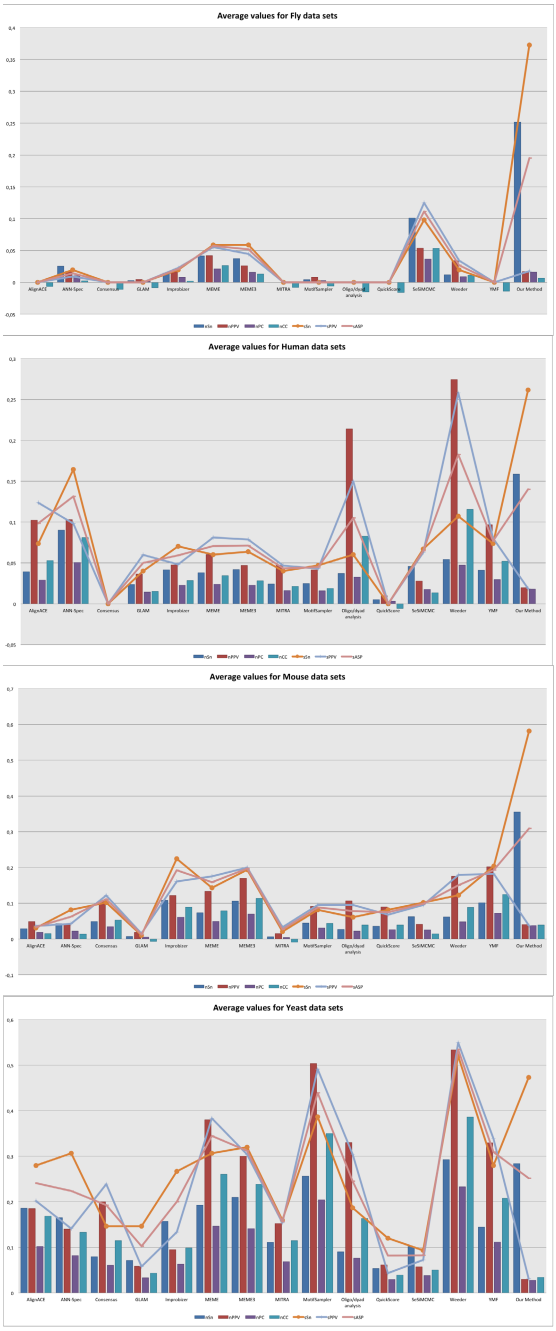

Fig. 3. Average statistical values depending on the organism

- Obviously, the underlying biology is yet to be completely understood, and therefore there is no standard method to measure the correctness of each tool in biological terms.
- Each one of the methods was tested by a different person, who made human choices for the parameters and the post processing of that method.
- The assessment only allows one predicted motif per data set (or none), even though the data sets of Type Real are most likely to contain several different binding sites.
- The length of the known motifs is in many cases longer than 30 bp, and our method, as well as most of the others, was run for motifs no longer than 12 bp.
- The assessment depends uniquely on TRANSFAC [8] for the definition of the known binding sites, and the TRANSFAC database might also contain errors.
- The method used to calculate the average of each tool tends to favor the methods that predict no motif for many data sets, as 0 is taken as the value for all the statistics in this case.
- The assessment was carried out in 2005, so it does not include methods developed in the last 10 years.

However, keeping all these in mind, the assessment serves as a powerful tool to infer some important conclusions about the performance of each method.

Our method shows really high values for three statistics: nSn, sSn, and $sASP$. But, on the other hand, the values for the statistics $nPPV$, nSp, and $sPPV$ are generally poor. From these, we can conclude that the method succeeds in predicting many of the sites, given the high number of true positives both at nucleotide level and site level, but lacks of a mechanism to detect false positives. This is understandable given the nature of the method. As it predicts sites according to statistics that measure the overrepresentation, it is very likely to happen that it reports many sites that are not actual instances of the motif but are very similar to it. Therefore, it usually finds the known motif, but with more instances than it actually contains.

As for the different organisms, it is interesting to notice that most of the other methods offer their best performance with yeast data sets, whereas our method gives its best results with fly and mouse data sets. There is no apparent reason for this, and it requires further investigation to figure out why this happens.

It is quite obvious that the main drawback of our method is the absence of a mechanism to detect false positives. For example, the method that gives the best overall statistics is Weeder. But this is, to a considerably extent, due to the fact that it was run in a cautious mode, predicting no motif in most of the cases. Our method, however, failed to detect the negative controls and predicted at least one motif for every given dataset, which produced a high number of false positives.

Even though there have been many different studies about DNA motif finding, it still remains as one of the most complicated challenges for researchers. Several different approaches have been recently developed and there has been

an important progress in this area. However, the task of comparing the performances of different motif finding tools have proven to be quite a struggle, given that each tool is designed based on an algorithm and on motif models that are too diverse and complex. This happens basically because we do not have yet a clear understanding of the biology of regulatory mechanisms. Therefore, it is not possible for us to define a standard to measure the quality of tools.

As many studies comparing the performance of different tools suggest [9] (and as researchers' experiments corroborate), the best option when trying to find motifs is using a few complementary tools in combination (selecting the top predicted motifs of each one), instead of simply relying on a single one.

According to this, we believe that our Statistical GA approach has proven to be suitable for being one of those complementary tools that can be used in addition to other ones to successfully predict motifs in any kind of set of DNA sequences. There is still work to do to improve the method, especially the addition of a mechanism to detect false positives, but we think that the method can be useful for researchers and that it might offer new ways for future development of computer-based motif finding methods.

References

1. Vijayvargiya, S., Shukla, P.: Identification of Transcription Factor Binding Sites in Biological Sequences Using Genetic Algorithm. International Journal of Research & Reviews in Computer Science 2(2) (2011)
2. Tanaka, E., Bailey, T. L., Keich, U.: Improving MEME via a two-tiered significance analysis. Bioinformatics, btu163 (2014)
3. Abnizova, I., te Boekhorst, R., Walter, K., Gilks, W.R.: Some statistical properties of regulatory DNA sequences, and their use in predicting regulatory regions in the Drosophila genome: the fluffy-tail test. BMC Bioinformatics 6(1), 109 (2005)
4. Shu, J.J., Li, Y.: A statistical thin-tail test of predicting regulatory regions in the Drosophila genome. Theoretical Biology and Medical Modelling 10(1), 11 (2013)
5. Tompa, M., Li, N., Bailey, T.L., Church, G.M., De Moor, B., Eskin, E., Favorov, A.V., Frith, M.C., Fu, Y., Kent, W.J., et al.: Assessing Computational Tools for the Discovery of Transcription Factor Binding Sites. Nat. Biotechnol. 23137–23147 (2005)
6. Pevzner, P.A., Sze, S.H.: Combinatorial approaches to finding subtle signals in DNA sequences. ISMB 8, 269–278 (2000)
7. Burset, M., Guigo, R.: Evaluation of gene structure prediction programs. Genomics 34(3), 353–367 (1996)
8. Wingender, E., Dietze, P., Karas, H., Knüppel, R.: TRANSFAC: a Database on transcription factors and their DNA binding sites. Nucleic Acids Res. 24, 238–241 (1996)
9. Das, M.K., Dai, H.K.: A survey of DNA motif finding algorithms. BMC Bioinformatics 8(Suppl. 7), S21 (2007)
10. Lenhard, B., Wasserman, W.W.: TFBS: Computational framework for transcription factor binding site analysis. Bioinformatics 18(8), 1135–1136 (2002)