International Journal of Computational Intelligence and Applications
Vol. 12, No. 3 (2013) 1350017 (15 pages)

DOI: 10.1142/S146902681350017X

Imperial College Press
www.icpress.co.uk

EFFICIENT DNA MOTIF DISCOVERY USING MODIFIED GENETIC ALGORITHM

ESSAM AL DAOUD
Computer Science Department
Zarqa University, Zarqa, Jordan
essamdz@zu.edu.jo

Received 30 January 2013
Revised 11 April 2013
Published 7 August 2013

In this study, a new genetic algorithm was developed to discover the best motifs in a set of DNA sequences. The main steps were: finding the potential positions in each sequence by using few voters (1−5 sequences), constructing the chromosomes from the potential positions, evaluating the fitness for each gene (position) and for each chromosome, calculating the new random distribution, and using the new distribution to generate the next generation. To verify the effectiveness of the proposed algorithm, several real and artificial datasets were used; the results are compared to the standard genetic algorithm, and Gibbs, MEME, and consensus algorithms. Although all the algorithms have low correlation with the correct motifs, the new algorithm exhibits higher accuracy, without sacrificing implementation time.

Keywords: Motif; DNA sequences; genetic algorithm; random distribution; sampling.

1. Introduction

Identification of motifs, which are relative short, has proved to be very important in order to understand better the functionality and heritability properties of DNA sequences.[1−4] Although many algorithms have been proposed in recent decades for discovering the motifs, it is still a very challenging problem, and the prediction accuracy is not satisfactory. Several factors make the motif problem difficult. The first is that the widths of the motifs are very short compared to the background; motif width is about,[5−14] while the preceding part of the DNA sequence is about 300−5000. The second is that motifs in DNA sequences are not exactly matched, but approximately matched. The third is that the optimal width of a motif is not known in advance. The fourth is that not all DNA sequences in a dataset contain the recommended motif. The fifth is that the nature stochastic process is not well understood. The sixth is that some parts of DNA sequences look like motifs, such as "AAAAAAAA" or "TGTGTGTG". In this paper, a new algorithm is developed for detecting motifs in DNA sequences, based on the genetic algorithm.[5−10,15] The new algorithm is able to achieve better accuracy rates than the previous algorithms.

The rest of this paper is arranged as follows: Section 2 introduces the related algorithms, such as Gibbs, MEME, consensus, and other probabilistic approaches. Section 3 describes the new genetic algorithm, where the initial generation of the chromosomes starts by some potential positions. Section 4 summarizes the datasets used to compare and test the new motifs. Section 5 tunes the genetic algorithm parameters. Section 6 introduces the experimental results of the new and previous algorithms, the discovered motifs, and the sample logos. Section 7 presents the complexity of the suggested algorithm and the running time of the critical stages.

2. Related Work

This section discusses four popular approaches for finding motifs: MEME algorithms, Gibbs sampling, consensus model, and the standard genetic algorithm.[11–14]

Let $\mathrm{DNA} = \{\mathrm{DNA}_1, \mathrm{DNA}_2, \ldots, \mathrm{DNA}_m\}$ be a set of m DNA sequences, each of length n. The MEME model assumes that there are two types of components; the nonmotif ("background") and motif positions in sequences, the parameters of the two components in DNA sequences, can be described by using the position probability matrix (PPM), as follows:

$$\theta = [\,\theta_0 \quad \theta_1\,] = \begin{bmatrix} P_{A,0} & P_{A,1} & P_{A,2} & \ldots & P_{A,W} \\ P_{T,0} & P_{T,1} & P_{T,2} & \ldots & P_{T,W} \\ P_{C,0} & P_{C,1} & P_{C,2} & \ldots & P_{C,W} \\ P_{G,0} & P_{G,1} & P_{G,2} & \ldots & P_{G,W} \end{bmatrix}, \tag{1}$$

where θ_1 is the motif parameters, θ_0 is the background component parameters, $P_{x,0}$ is the probability of nucleotide x found at the background position, and $P_{x,i}$ is the probability of nucleotide x found at the ith motif position ($1 \le i \le w$). MEME maximizes the expectation of the joint likelihood by repeating the following two main steps:

Step 1: Compute (Step E)

$$Z^t = \underset{(Z|\mathrm{DNA},\theta^t)}{E}[Z] \tag{2}$$

Step 2: Solve (Step M)

$$\theta^{t+1} = \max_{\theta} \underset{(Z|\mathrm{DNA},\theta^t)}{E}[\log p(\mathrm{DNA}, Z|\theta)], \tag{3}$$

where $Z = \{Z_{i,j} \,|\, 1 \le i \le m, 1 \le j \le n\}$

$$Z_{i,j} = \begin{cases} 1 & \text{motif starts at } \mathrm{DNA}_{i,j} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

and

$$\log p(\mathrm{DNA}, Z \,|\, \theta) = \sum_{i=1}^{m} \sum_{j=1}^{n} Z_{i,j} \log p(\mathrm{DNA}_i Z_{i,j} = 1, \theta) + m \log \frac{1}{n}. \tag{5}$$

The likelihood of the ZOOPS model can be calculated by adding a new parameter γ, as follows:

$$\begin{aligned}\log p(\text{DNA}, Z \mid \theta, \gamma) = & \sum_{i=1}^{m} \sum_{j=1}^{n} Z_{i,j} \log p(\text{DNA}_i \mid Z_{i,j} = 1, \theta) \\ & + \sum_{i=1}^{m} (1 - Q_i) \log p(\text{DNA}_i \mid Q_i = 0, \theta) \\ & + \sum_{i=1}^{m} (1 - Q_i) \log(1 - \gamma) + Q_i \log \lambda, \end{aligned} \tag{6}$$

where γ is the prior probability of motif occurrence and λ is the prior probability of motif start position.The likelihood for the TCM model is:

$$\begin{aligned}\log p(\text{DNA}, Z \mid \theta, \lambda) = & \sum_{i=1}^{m} \sum_{j=1}^{n} (1 - Z_{i,j}) \log p(\text{DNA}_{i,j} \mid \theta_0) \\ & + Z_{i,j} \log p(\text{DNA}_{i,j} \mid \theta_1) \\ & + (1 - Z_{i,j}) \log(1 - \lambda) + Z_{i,j} \log \lambda. \end{aligned} \tag{7}$$

Gibbs sampling approximates the joint probability distribution by generating a sequence of samples from the conditional distribution of each variable. Gibbs sampling maximizes the log-odds likelihood ratio as follows:

$$\text{Maximize} \sum_{i=1}^{4} \sum_{j=1}^{w} c_{i,j} \log \frac{q_{i,j}}{p_i}, \tag{8}$$

where $c_{i,j}$ is the count of the letter i in this position (the matrix here is the position-specific scoring matrix PSSM),

$$q_{i,j} = \frac{c_{i,j} + b_i}{m - 1 + B} \tag{9}$$

and

$$p_i = \frac{c_{0,i} + b_i}{S + B} \tag{10}$$

m is the number of the DNA sequences, B pseudo-counts to avoid zeros, and S is the sum of observed counts of all letters in the background. However, Gibbs sampling maximizes Eq. (8) iteratively by using three steps:

Step 1: Predictive update, choose one of m sequences randomly, find position-specific scoring matrix PSSM, and calculate $q_{i,j}$ and p_i as described in Eqs. (9) and (10).

Step 2: Calculate new distribution

$$R_i = Q_i / P_i$$

and

$$A_i = R_i \Big/ \sum_{i=1}^{n-w+1} R_i \quad i = 1, 2, \ldots, n-w+1,$$

where

$$Q_i = Pr(\text{word} \,|\, \text{motif}) = PSSM_{x_i,1} \times \cdots \times PSSM_{x_{i+w-1},w}$$
$$P_i = Pr(\text{word} \,|\, \text{Background}) = PSSM_{x_i,0} \times \cdots \times PSSM_{x_{i+w-1},0}$$

Step 3: Sampling step; sample a random position based on the new distribution in step 2.

The main idea in Gibbs sampling is that a more accurate selection in step 1 leads to more accuracy in step 3, and vice versa.[16]

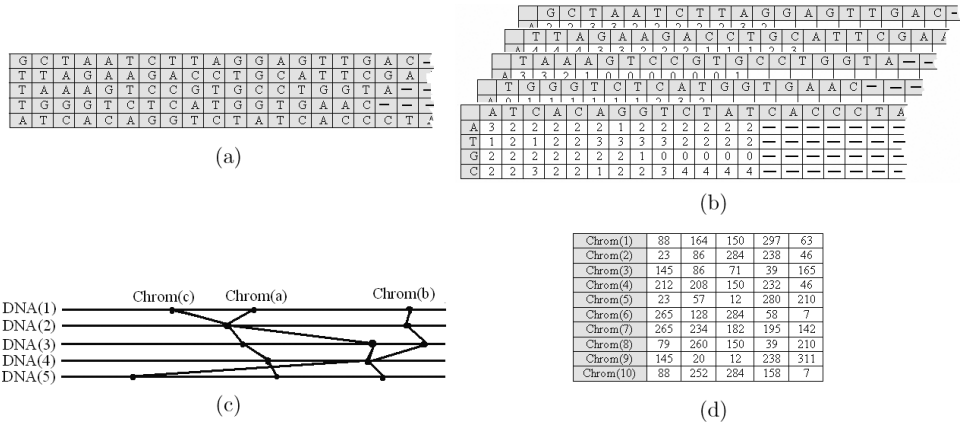
The consensus algorithm uses a greedy algorithm and the weight matrices to maximize the information content. The general strategy in this model is to find a pair of DNA sequences with the highest information content and share a motif; the next step is to find a third sequence and preserve the previous motif; and so on.[17,18]

The last algorithm discussed here is the previous genetic algorithm used to find motifs. In Ref. 19, the authors represent chromosomes as binary strings where each 16 bits represent one random start position. The fitness is the information content, which is extracted from the position-specific scoring matrix PSSM. The authors used one- and two-point crossovers and a bitwise mutation operation, in which both mutation and crossover occur according to a specific rate.[20]

3. New Genetic Algorithm

The main idea in the new genetic algorithm is to focus on some potential positions in each sequence by using t voters (sequences), and then constructing k chromosomes, where each position is considered as one gene. Thus, the number of genes in one chromosome is equal to the number of processed sequences. Unlike the standard approach, the new algorithm generates a new child from all the chromosomes, where each gene in the child chromosome is selected based on a new random distribution. Two levels of fitness functions are applied. The first is to evaluate each gene (by Algorithm 5); we used it to construct the new random distribution. The second level is to evaluate each chromosome; we use it to rank the motifs and to stop the procedure. Algorithm 1 describes the basic steps. We will consider the following notation in the next algorithms:

m: number of sequences
n: length of the sequence
k: number of chromosomes
s: highest s positions
v: number of letters (four letters for DNA sequences)
t: number of voters



Chrom(1)	88	164	150	297	63
Chrom(2)	23	86	284	238	46
Chrom(3)	145	86	71	39	165
Chrom(4)	212	208	150	232	46
Chrom(5)	23	57	12	280	210
Chrom(6)	265	128	284	58	7
Chrom(7)	265	234	182	195	142
Chrom(8)	79	260	150	39	210
Chrom(9)	145	20	12	238	311
Chrom(10)	88	252	284	158	7

Fig. 1. (a) Sample DNA sequences, (b) The frequency matrix A, (c) Sample chromosomes that are constructed randomly, (d) Ten chromosomes of length 5.

z: number of mutations
w: width of the motif
DNA$[1..m]$: array of DNA sequences [see Fig. 1(a)].
$A(1..m, 1..n, 1..4)$: frequency of each letter (A, T, G and C) in the next w characters [see Fig. 1(b)].
Pos$(1..m, 1..s)$: to store the best s position for each DNA sequence.
Chrom$(1..k, 1..m)$: to store k chromosomes, each one consisting of k random positions [see Figs. 1(c) and 1(d)].
Count$(1..m, 1..k)$: to store the number of similar motifs at each position.
Prob$(1..n, 1..m)$: to store the probability based on the *count* for each position

Algorithm 1: New genetic algorithm
Input: DNA sequences and w
Output: the best motifs

Step 1: Count and store the frequency of the letters of each position and sequence (call Algorithm 2).
Step 2: Select the best s positions (potential motifs) in each DNA sequence by calling Algorithm 3.
Step 3: Construct k chromosomes by selecting one potential position from each DNA sequence randomly; thus, the length of the chromosome is m (call Algorithm 4).
Step 4: Evaluate each gene (motif) in the chromosomes by calling Algorithm 5 (fitness algorithm).
Step 5: Normalize the values in step 4 and convert them to probabilities, such that the probability sum of each k motif in any sequence is one (call Algorithm 6).

Step 6: Draw new k chromosomes from the new probability distribution (see algorithm 7) (crossover).
Step 7: Go to step 4 until converge.

Algorithm 2: Counting
Input: motif length w and m DNA sequences
Output: $\mathbf{A}(m, n, 4)$, the frequency matrix of the DNA letters at each position and length w

```
For i = 1 to m
  For j = 1 to length of DNA( i )- w+1
    c = (j > w?w − 1 : j − 1)
      for k = 0 to c
        A(i, j-k, DNA(i, j) )++
        // where the characters A, T, G and C are replaced by 1, 2, 3, and
             4 respectively
Return A
```

Algorithm 3: best positions
Input: matrix $\mathbf{A}$, integer number s, motif length w, t and m DNA sequences
Output: best s positions for each DNA sequence stored in $\mathbf{Pos}(m, s)$

For $x = 1$ to t
  For $i = 1$ to m
    $b = \text{random}(1..m)$
    if $b = i$ then continue
    For $j = 1$ to DNA(i)
      $B(i, j) = B(i, j)+$

$$\max_{c=1\ldots \text{DNA}[b]} \left\{ \sum_{v=1}^{4} \frac{2w - |A(i, c, v) - A(b, j, v)|}{2} \right\} \tag{11}$$

    $\mathbf{Pos}(i, 1..s) = \text{The highest } s \text{ integers in B}(i, 1 \ldots \text{DNA}(i))$

Algorithm 4: Chromosome construction
Input: matrix **POS**, k, n, and m
Output: **Chrom**(k, m), k chromosome with m length

For $i = 1$ to k
  For $j = 1$ to m
    Chrom $(i, j) =$ **Pos**(j, $\text{Rand}_{Uni}(1..s)$)

Algorithm 5: motif evaluation (fitness algorithm)
Input: $\mathbf{A}(m, n, v)$, $\mathbf{Chrom}(k, m)$, m, w, and k

Output: $\mathbf{Count}(m, k)$

For $i = 1$ to m
For $j = 1$ to k

$$\mathbf{Count}(j, i) = \sum_{\substack{u=1 \\ u \neq i}}^{m} \underset{l=w}{ED}(\text{Chrom}(j, u), \text{Chrom}(j, i)), \tag{12}$$

where ED is bit-parallel approximate string matching of length w[21–25]; the meaning of the above expression is the sum of the edit distance between the current substring and all the other substrings that start at the given position, and its length is w.

Algorithm 6: Normalization
Input: $\mathbf{Count}(m, k)$
Output: $\mathbf{Prob}(m, k)$

$\mathbf{Prob}(1..m, 1..k) = 0$
For $i = 1$ to m
$D(i) = \sum_{x=1}^{k} \text{Count}(x, i)$ the summation is applied to the distinct values in $\text{Count}(x, i)$ for all $x = 1..k$.
For $j = 1$ to k

$$\mathbf{Prob}(i, \mathbf{chrom}(j, i)) = \frac{\text{Count}(j, i)}{D(i)} \tag{13}$$

Let some of zero elements in $\mathbf{Prob}(i, 1..n) =$ minimum of nonzero elements in $\mathbf{Prob}(i, 1..k)$ and renormalize the result; the number of replaced elements is denoted by z (mutation). We store in $\mathbf{Prob}(i, \mathbf{Chrom}(j, i))$ the higher value if one position has more than one value.

Algorithm 7: New chromosomes (crossover)
Input: $\mathbf{Pos}(m, k)$, $\mathbf{Prob}(m, n)$, number of chromosomes k, and chromosome length m
Output: $\mathbf{Chrom}(k, m)$

For $i = 1$ to k
For $j = 1$ to m
$\mathbf{Chrom}(i, j) = \mathbf{Pos}(j, \text{Rand}_{\Omega}(j))$
where the jth random distribution is generated by $\mathbf{Prob}(j, 1..n)$ and denoted by $\text{Rand}_{\Omega}(j)$ for $j = 1..m$

Convergence condition is $\text{min\_Chorm}_t = \text{min\_Chorm}_{t-1}$ for some iterations where min_Chorm_t is

$$\min_{i=1...k} \left(\sum_{j=0}^{m} \text{Prob}(j, \text{chrom}(i, j)) \right) \text{ in iteration } t. \tag{14}$$

Table 1. Sample dataset from TRANSFAC together with binding sequences (Motifs).

Name	TFAC	m	Motif	Motif Len. (w)	Avg. Length
AFT2	RCS1	50	GGGTGC	6	443.86
BAS1	BAS1	12	TGACTC	6	419.77
CAD1	CAD1	20	ATTAGTAAGC	10	553.04
CBF1	CBF1	100	TCACGTG	7	597.80
DAL82	DAL82	50	GATAAG	6	547.08
DIG1	STE12	50	TGAAACA	7	617.03
GAT1	GZF3	40	AGATAAG	7	516.02
PHO4	PHO4	20	CACGTGG	7	566.48
REB1	REB1	40	CGGGTAA	7	440.87
STE12	STE12	100	TGAAACA	7	606.49
TYE7	CBF1	50	TCACGTGAT	9	574.29

We can rank the winner chromosome with regard to motif width by using the following formula:

$$\text{Ranking} \sim \left(\sum_{j=0}^{m} \text{Count}(\text{chrom}(\text{winner}, j), j)\right) \Big/ (wm^2) \tag{15}$$

4. Dataset

Table 1 shows several sets of eukaryotic DNA sequences that can be downloaded from TRANSFAC® (Refs. 26 and 27) in FASTA format, together with consensus binding sequences (motifs). The transcription factors have been proven experimentally to act in a synergistic or antagonistic manner. TRANSFAC® is based on ChIP-ChIP and ChIP-Seq datasets and provides information about transcription factors and regulated genes. Table 2 shows an artificial dataset generated randomly according to a Markov chain of order 3; the dataset can be downloaded from Ref. 28.

5. Parameters Tuning

To assess the previous algorithms and the new algorithm, we used the statistics introduced in Refs. 28–30, where two levels are used, at the nucleotide level and at

Table 2. Sample artificial dataset generated randomly according to a Markov chain of order 3.

Name	Used Name	# Groups	# Sequences (m)	Seq. Length (w)
Fly	Dm01m-08m	8	1–5	∼1000–2000
human	Hum01m-26m	26	3–17	∼400–2000
Mouse	Mus01m-12m	12	3–12	∼400–1500
Yeast	Yst01m-10m	10	3–12	∼400–1500

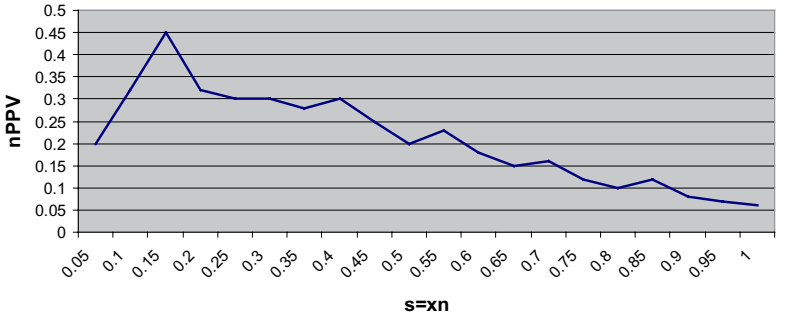


Fig. 2. Number of the potential positions versus nPPV.

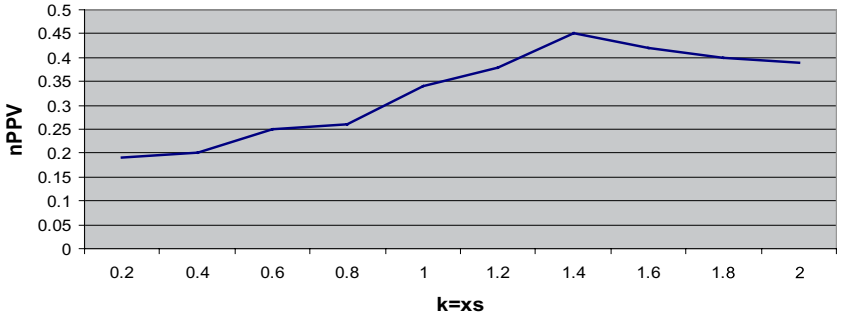


Fig. 3. Number of the chromosomes versus nPPV.

the site level. The statistics in the first level can be described as follows:

- nTP: number of common positions in the predicted and known motifs
- nFN: number of nucleotides in the known motifs minus the common positions
- nFP: number of nucleotides in the predicted motifs minus the common positions
- nTN: number of nucleotides not in the predicted motifs or in the known motifs

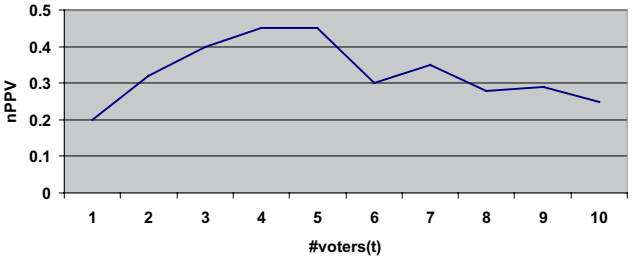


Fig. 4. Number of voters versus nPPV.

The same statistics can be used in the site level as follows:

- sTP: number of overlapped motifs (predicted motif overlaps by at least 1/4 of the known motif)
- sFN: number of known motifs minus the overlapped motifs
- sFP: number of predicted motifs minus the overlapped motifs
- sTN: number of sites excluding the predicted motifs and the known motifs.

Thus, we can define the sensitivity and the positive predictive value as follows:
Nucleotide level sensitivity:

$$nSn = \frac{nTP}{nTP + nFN}. \tag{16}$$

Nucleotide level positive predictive value:

$$nPPV = \frac{nTP}{nTP + nFP}. \tag{17}$$

Site level sensitivity:

$$sSn = \frac{sTP}{sTP + sFN}. \tag{18}$$

Site level positive predictive value:

$$sPPV = \frac{sTP}{sTP + sFP}. \tag{19}$$

Before applying the new algorithm to the complete datasets, a small random dataset was selected and used to tune the new genetic algorithm parameters, namely, the number of potential positions (s) with respect to the sequence length, the number of chromosomes (k), the number of voters in each sequence (t), and the number of mutations (z). We found that after a fixed number of iterations, the best parameters were $s = 0.15\,\mathrm{n}$, $k = 1.4\,\mathrm{s}$, $t = 4$ sequences, and $z = n/100$ for nucleotide level positive predictive value (nPPV), as shown in Figs. 2–4. Similar results were obtained by using the other statistics.

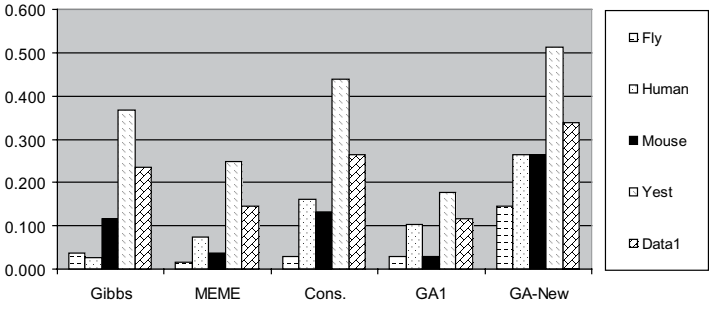


Fig. 5. Nucleotide level sensitivity (nSn) for all methods and all datasets.

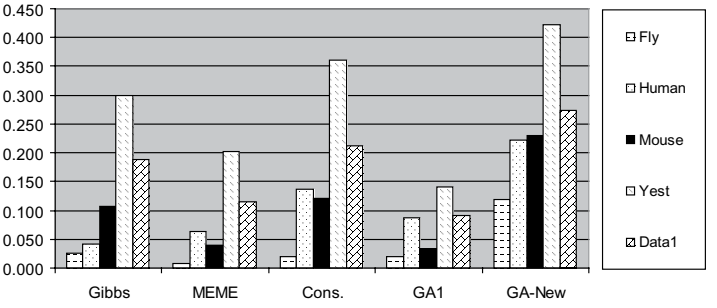


Fig. 6. Nucleotide level positive predictive value (nPPV) for all methods and all datasets.

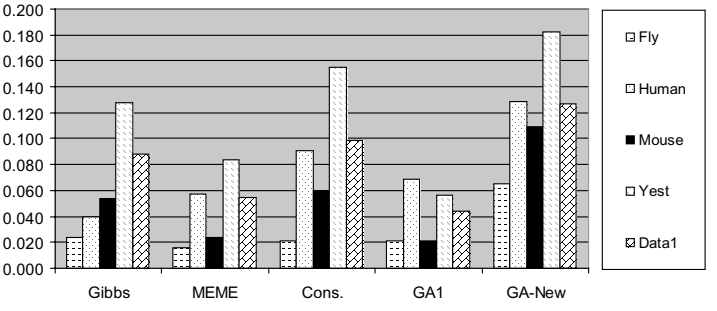


Fig. 7. Site level sensitivity (sSn) for all methods and all datasets.

6. Experimental Results

The new algorithm was compared with four algorithms: Gibbs, MEME, consensus and genetic algorithm in Ref. 19. All the algorithms were reimplemented by our platform, using a 2.3 GHz PC, Windows XP and Java JDK7. Figures 5–8 show that all the algorithms, including the suggested algorithm, have low correlation with the motifs introduced by TRANSFAC®. However, the suggested algorithm exhibits better performance in terms of accuracy and implementation time. For example, the

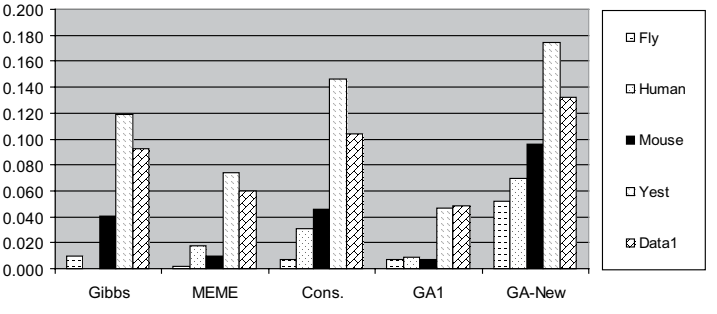


Fig. 8. Site level positive predictive value (sPPV) for all methods and all datasets.

Fig. 9. Motif logo representation of YST03M, CBF1, DIG1 and STE12.

results of the nucleotide level sensitivity (nSn) using the new algorithm were 0.15, 0.25, 0.25, 0.52 and 0.33 for the datasets fly, human, mouse, yeast and Data1 (the dataset in Table 1), respectively. The next accurate algorithm according, to our implementation, was the consensus algorithm. The same indication can be noted if the other statistics are used.

Table 3 shows the motifs discovered by using the new algorithm for each DNA sequence in Table 1 and sample DNA sequences from Table 2; the highest three motifs were selected, and relative ranks assigned to them, calculated by Eq. (15).

Figure 9 show a sample logo representation of the discovered motifs. All logos were generated by WebLogo (http://weblogo.berkeley.edu/). These figures illustrate that the suggested algorithm can identify new motifs efficiently.

Table 3. Motifs Discovered by the new algorithm, ranks and width for the dataset in Tables 1 and 2.

Name	Motif	Rank	Len. (w)	Name	Motif	Rank	Len. (w)
AFT2	CGATCGG	2	7	TYE7	TTTCAAAA	1	8
	CGGGGGT	1	7		TATTATC	3	7
	TATGTAGA	3	8		TGTATTATCA	2	10
BAS1	TATATAA	3	7	Dm03m	GAAAAAC	2	7
	ATATAA	1	6		TTTTACCCT	1	9
	TATAAA	2	6		TTACCCTTG	3	9
CAD1	TTTTTTTA	1	8	Dm04m	TAAAAACTT	1	9
	AAGAAAT	3	7		AAATTAATT	2	9
	TTGAAAAA	2	8		TTTGAAGAG	3	9
CBF1	TCACGTGAC	3	9	Hm04m	GTAAGGCGG	3	9
	TCACGTG	2	7		CCGCGCGGG	1	9
	CACGTGA	1	7		TTTCCTCC	2	8
DAL82	AAGAAAA	1	7	Hm08m	GCCGCCT	2	7
	AGAAAAA	2	7		CTCCCCCCA	3	9
	GAAAAAAA	3	8		GGCCGCC	1	7
DIG1	GAAAAATGA	2	9	Mus10m	AAGAAACA	1	8
	AATGAAACAA	1	10		GGCGGGAGA	2	9
	ATCTTTTA	3	8		TTTCCGCTT	3	9
GAT1	TAGAGAT	1	7	Mus11m	GGGGGAGG	1	8
	ACGAAAA	2	7		GGGGGAGGC	2	9
	AATAAGT	3	7		GGGGAGGC	3	8

Table 3. (*Continued*)

Name	Motif	Rank	Len. (w)	Name	Motif	Rank	Len. (w)
PHO4	CCACGTG	1	7	Yst01m	AGAGATAAA	3	9
	GTAATAAAAA	3	10		ATATTTTTT	1	9
	TAATAAAAAT	2	10		TAAAAAA	2	7
REB1	TGTAGGG	2	7	Yst03m	TTTGACTCA	1	9
	TTATTACTT	1	9		TCTTGTA7	2	7
	TAATGATAT	3	9		TGTAATT7	3	7
STE12	TGAAACA	1	7	Dm03m	GAAAAAC	2	7
	AAAATGAA	2	8		TTTTACCCT	1	9
	TGAAACAA	3	8		TTACCCTTG	3	9

7. Complexity and Time

The most costly sub-algorithms were Algorithms 3 and 5. The worst case of Algorithm 3 was $v \times t \times m \times n^2$. In this study, as mentioned previously, v and t are fixed to four. Thus, the complexity of Algorithm 3 is $\theta(mn^2)$ comparisons. While the worst case of evaluating one chromosome by using bit-parallel approximate string matching is $dw^2 \sum_{i=1}^{m} i = dw^2 m(m+1)/2$, where d is the expected number of characters mismatch and w is the motif length, subsequence, the complexity of evaluating one chromosome is $\theta(m^2)$, and the complexity of finding one generation (Algorithm 5) is $\theta(\mathrm{km}^2)$. Table 4 illustrates the average length of the datasets, the required time to implement Algorithm 3, the required time to implement one generation (Algorithm 5), the number of generations, and the required time to implement all the generations. We can note that Algorithm 3 needs more time than the other algorithms. For example, the implementation time of Algorithm 3 on the dataset CBF1 was about 571.7 ms, while the required time to process all the generations was 54.9 ms, and the total time was 626.7 ms. However, the required times to implement the MEME and Gibbs algorithms on the same dataset were about 905 ms and 1022 ms, respectively.

Table 4. Sample implementation time of Algorithm 3 in milliseconds.

Name	m	Average n	Alg 3	One Generation	Number of Generations	All Generations	Total
CBF1	100	597.80	571.78	1.6160	33	54.944	626.73
DAL82	50	547.08	239.44	0.2040	37	7.956	247.39
DIG1	50	617.03	304.58	0.2040	40	6.120	310.70
STE12	100	606.49	588.53	1.6160	25	45.248	633.78
TYE7	50	574.29	263.85	0.2040	23	4.488	268.34
Dm04m	5	1604.0	205.83	0.0002	44	0.011	205.84
Hm08m	16	473.75	57.46	0.0070	39	0.258	57.71
Mus10m	14	933.42	195.17	0.0047	59	0.259	195.42
Yst03m	9	448.88	29.02	0.0013	38	0.039	29.05

8. Conclusion

In this study, we have suggested a new direction for discovering and ranking motifs. The new genetic algorithm finds the minimum edit distance by using voters and then constructing a new random distribution to minimize the total edit distance. Using a large variety of real and artificial datasets generated randomly according to a Markov chain of order 3 proved that the new algorithm provides a better accuracy and implementation time than the previous algorithms. This study can be extended to RNA and protein sequences. Moreover, some conditions can be added to exclude the repeaters.

References

1. G. D. Stormo, DNA binding sites: Representation and discovery, *Bioinformatics* **16** (2000) 16−23.
2. H. D. Huang, J. T. Horng, Y. M. Sun, A. P. Tsou and S. L. Huang, Identifying transcriptional regulatory sites in the human genome using an integrated system, *Nucl. Acids Res.* **32** (2004) 1948−1956.
3. S. Sinha, M. Blanchette and M. Tompa, PhyME: A probabilistic algorithm for finding motifs in sets of orthologous sequences, *BMC Bioinf.* **5** (2004) 170.
4. A. Sandelin and W.W. Wasserman, Constrained binding site diversity within families of transcription factors enhances pattern discovery bioinformatics, *J. Mol. Biol.* **338** (2004) 207−215.
5. C. Kingsford, E. Zaslavsky and M. Singh, A compact mathematical programming formulation for DNA motif finding, *Lect. Notes Comput. Sci.* **4009** (2006) 233−245.
6. D. Liu, X. Xiong, B. DasGupta and H. Zhang, Motif discoveries in unaligned molecular sequences using self-organizing neural network, *IEEE Trans. Neural Netw.* **17** (2006) 919−928.
7. K. Shida, GibbsST: A Gibbs sampling method for motif discovery with enhanced resistance to local optima, *BMC Bioinf.* **7** (2006) 486.
8. F. Fauteux, M. Blanchette and M. V. V. Strömvik, Seeder: Discriminative seeding DNA motif discovery, *Bioinformatics* **24**(20) (2008) 2303−2307.
9. M. Federico, P. Valente, M. Leoncini, M. Montangero and R. Cavicchioli, An efficient algorithm for planted structured motif extraction, in *Proc. First ACM Workshop on Breaking Frontiers of Computational Biology* (2009), pp. 1−6.
10. Marschall and S. Rahmann, Speeding up exact motif discovery by bounding the expected clump size, in *Proc. 10th International Workshop on Algorithms in Bioinformatics (WABI)* (2010), pp. 337−349.
11. H. Sun *et al.*, Tmod: Toolbox of motif discovery, *Bioinformatics* **26**(3) (2010) 405−407.
12. C. Bi, A Monte Carlo EM algorithm for de novo motif discovery in biomolecular sequences, *IEEE/ACM Trans. Comput. Biol. Bioinf.* **6**(3) (2009) 370−386.
13. C. Chen, B. Schmidt, L. Weiguo and W. M. Uller-Wittig, GPU-MEME: Using graphics hardware to accelerate motif finding in DNA sequences, in *Proc. Third IAPR Int. Conf. Pattern Recognition in Bioinformatics (PRIB)* (2008), pp. 448−459.
14. M. Defrance and J. van Helden, Info-gibbs: A motif discovery algorithm that directly optimizes information content during sampling, *Bioinformatics* **25**(20) (2009) 2715−2722.
15. S. Sinha and M. Tompa, YMF: A program for discovery of novel transcription factor binding sites by statistical over representation, *Nucl. Acids Res* **31** (2003) 3586−3588.

16. G. D. Stormo, Motif discovery using expectation maximization and gibbs' sampling, 2010 *Meth. Mol. Biol.* **674** (2010) 85−95.
17. G. Z. Hertz and G. D. Stormo, Identifying DNA and protein patterns with statistically significant alignments of multiple sequences, *Bioinformatics* **15**(7−8) (1999) 563−577.
18. G. D. Stormo, Maximally efficient modeling of DNA sequence motifs at all levels of complexity, *Genetics* **187**(4) (2011) 1219−1224.
19. D. Che, Y. Song and K. Rasheed, MDGA: Motif discovery using a genetic algorithm, *Genetic and Evolutionary Computation Conf.* (GECCO2005), pp. 447−452.
20. L. X. Sean and W. Dianhui, an improved genetic algorithm for DNA motif discovery with public domain information, *Adv. Neuro-Inf. Process. Lect. Notes Comput. Sci.* **5506** (2009) 521−528.
21. G. Navarro and R. Baeza-Yates, Improving an algorithm for approximate pattern matching, *Algorithmica* **30**(4) (2001) 473−502.
22. A. Apostolico and C. Tagliacollo, Incremental discovery of the irredundant motif bases for all suffixes of a string in $o(n^2 \log n)$ time, *Theor. Comput. Sci.* **408**(2−3) (2008) 106−115.
23. B. Ma and X. Sun, More efficient algorithms for closest string and substring problems, *SIAM J. Comput.* **39**(4) (2009) 1432−1443.
24. G. Navarro, A guided tour to approximate string matching, *ACM Comput. Surveys* **33**(1) (2001) 31−88.
25. M. Li, B. Ma and L. Wang, On the closest string and substring problems, *J. ACM* **49** (2002) 157−171.
26. O. V. Kel-Margoulis, A. E. Kel, I. Reuter, I. V. Deineko and E. Wingender, TRANS-Compel®: A database on composite regulatory elements in eukaryotic genes, *Nucl. Acids Res.* **30** (2002) 332−334.
27. V. Matys, E. Fricke, R. Geffers, E. Gling, M. Haubrock, R. Hehl, K. Hornischer, D. Karas, A. E. Kel, O.V. Kel-Margoulis, D. U. Kloos, S. Land, B. Lewicki-Potapov, H. Michael, R. Mnch, I. Reuter, S. Rotert, H. Saxel, M. Scheer, S. Thiele and E. Wingender, TRANS-FAC: Transcriptional regulation, from patterns to profiles, *Nucl. Acids Res.* **31** (2003) 374−378.
28. M. Tompa, N. Li, T. L. Bailey, G.M. Church, B. De Moor, E. Eskin, A. V. Favorov, M. C. Frith, Y. Fu, W. J. Kent *et al.*, Assessing computational tools for the discovery of transcription factor binding sites, *Nat. Biotechnol.* **23** (2005) 137−144.
29. E. S. Jackson and W. J. Fitzgerald, A sequential Monte Carlo EM approach to the transcription factor binding site identification problem, *Bioinformatics* **23**(10) (2007) 1313−1320.
30. L. Li, R. L. Bass and Y. Liang, fdrMotif: Identifying cis-elements by an EM algorithm coupled with false discovery rate control, *Bioinformatics* **24**(5) (2008) 629−636.