Natural Computing (2019) 18:333–350
https://doi.org/10.1007/s11047-018-9674-1

Discovery of conserved regions in DNA sequences by Artificial Bee Colony (ABC) algorithm based methods

Dervis Karaboga[1] · Selcuk Aslan[2]

Published online: 24 February 2018

Abstract
The conserved regions between genomic sequences extracted from the different species give important information about the complex regulatory mechanisms of the transcription and translation processes. However, identification of these regulatory regions that are short DNA segments and called motifs is a major challenge in bioinformatics. With the avalanche of the newly sequenced genomic data and our evolving understanding of the characteristics of the regulatory mechanisms, there is still a need for developing fast and accurate motif discovery techniques or considerably refinement on the existing models. In this paper, we presented two different Artificial Bee Colony algorithm based motif discovery techniques and investigated their serial and parallelized implementations. Experimental studies on the three real data sets showed that the proposed methods outperformed other metaheuristics in terms of similarity values of the predicted motifs.

Keywords Motif discovery · Metaheuristics · Artificial Bee Colony algorithm

1 Introduction

The genomic era, started with the publication of complete sequences of some model organisms including mouse, chimpanzee and also human, has evolved to a new situation in which development of new algorithms or adaptation of known computational techniques to catalog and locate the entire set of functional elements regulating complex interactions between protein and deoxyribonucleic acid (DNA) molecules are the main concern. One of the most important functional elements encoded in the genomic data is transcription-factor binding sites to which the proteins known as transcription factor bind. Transcription-factor binding sites are short residue sequences and can be defined as 5–30 nucleotide long motifs or words (Bulyk 2004; Tompa 2005; Li and Tompa 2006; Das and Dai 2007). Exact location of the transcription-factor binding sites can be vary within a promoter, and especially in eukaryotic genomes they can be found downstream or upstream of the intron of the gene being regulated. In addition to this, they are frequently degenerate which means that more than one possible nucleotide sequence can be used to represent a binding site for higher genomes and much of the information about them comes from the traditional methodologies that are time-consuming and require considerable amount of efforts. In response to this challenge, computational prediction approaches have been presented in recent years that predict the start position and nucleotide sequence of the mentioned motifs.

Based on the type of combinatorial methods employed by the algorithms, the motif discovery or finding techniques can be categorized into two major groups (Bulyk 2004; Tompa 2005; Li and Tompa 2006; Das and Dai 2007). In the first group of algorithms, the motifs are tried to be found by utilizing exhaustive counting and comparing techniques. While the exhaustive enumeration methods guarantee the finding of the most appropriate motifs and their implementations with the specialized data structures such as suffix trees increase their usability with the problems that need to find short motifs, conservation level of the motifs can be important and directly affect the

✉ Selcuk Aslan
selcuk.aslan@omu.edu.tr

Dervis Karaboga
karaboga@erciyes.edu.tr

1 Department of Computer Engineering, Erciyes University, Kayseri, Turkey

2 Department of Computer Engineering, Ondokuz Mayis University, Samsun, Turkey

produced results (Bulyk 2004; Tompa 2005; Li and Tompa 2006; Das and Dai 2007). In the second group of the algorithms, the methods use sequence or motif models that are estimated by taking advantages of sound maximum-likelihood procedures or Bayesian inference. The Expectation Maximization (EM) introduced by Lawrence and Reilly and Gibbs Sampling that is also proposed by Lawrence et al. (1990) are effective statistical techniques that are main motivation of the popular motif discovery techniques such as MEME and Gibbs sampler. In recent years, evolutionary computation techniques have attracted the researchers attention and they have been used to solve complex bioinformatics problems including alignment of sequences (Ozturk and Aslan 2016), structure prediction (Liu 2013; Li 2015), protein–protein interaction (Cao 2015) and classification (Martinez 2010; Alshamlan 2015). Liu et al. (2004) introduced a Genetic Algorithm (GA) based technique called FMGA for motif discovery problem (Liu). In FMGA, mutation and crossover operators were specialized for the motif discovery problem. When mutation operator was applied to an individual in the population, in order to protect highly conserved regions, position weight matrices were used Liu (2004). A similar procedure was also applied on the crossover operation. Special gap arrangement and penalized procedures were employed for increasing the quality of the offspring. Particle Swarm Optimization (PSO) algorithm based technique for detecting motifs in amino acid sequences was presented by Chang et al. (2004) and they achieved better true positive hit when compared to the motifs reported in the PROSITE. Another GA based motif discovery technique called MDGA was proposed by Che (2005). Experimental studies for MDGA showed that this technique produced more accurate motifs compared to the Gibbs sampling (Che 2005). Shao and Chen (2009) proposed a new Bacterial Foraging Optimization (BFO) algorithm combined with the memory or tabu list part of the Tabu Search (TS) to guide the search optimization. Huo et al. (2010) combined GA and Random Projection Strategy (RPS) to identify l nucleotide length motifs in the sequences with the maximum d substitutions. Chan et al. (2012) used Memetic Algorithm (MA) with various algorithm components and models for motif discovery problem. Lia-wei and Ting utilized Immune Genetic Algorithm, that is a GA based algorithm combined with immune mechanism, and demonstrated the usability of the evolutionary computation methods to find motifs in relatively long promoter sequences Luo and Wang (2010).

Multiobjective adaptation of these algorithms is another important topic in the study of motif discovery. Kaya (2009) proposed a GA based multiobjective technique in which some type of conflicting arguments including number of sequences used to generate consensus sequence, length of motifs and finally their similarity values are tried to be optimized simultaneously. Gonzalez-Alvarez et al. (2010) used Differential Evolution (DE) algorithm with pareto tournaments. They also proposed a multiobjective Artificial Bee Colony (ABC) algorithm with genetic operator in order to address the multiobjective optimization of motif discovery problem and analyzed its performance on different multiobjective adaptation including multi-term fitness function, ranking and sorting methodologies (González-Álvarez 2011, 2012). Karaboga and Aslan (2016a) presented an ABC based motif discovery technique named consensusABC in which the candidate generation is directed with the consensus string of the alignment by taking number of sequences and length of motif being discovered as constants. However, before modeling motif discovery problem as a multiobjective optimization problem and using a multiobjective implementation of the evolutionary computational techniques that are originally proposed to solve numerical optimization problems, we must correctly adapt their distinctive steps to the considered discrete problems. In this study, we described two new discrete ABC algorithms by adhering to the individual effect of the candidate generation approach in the mathematical expression used in the employed and onlooker bee phases. In addition to this, ABC algorithm based models were parallelized depending on the ring migration topology. Experimental studies on the three different data sets extracted from the TRANSFAC database (Matys 2003) showed that both serial and parallel implementations of the proposed ABC algorithm based models outperformed other metaheuristic techniques used in the comparison. The rest of the paper is organized as follows. In Sect. 2, formal definition of the motif discovery problem and related terminology are summarized. Fundamental steps of the ABC algorithm and its adaptation to the motif discovery problem are given in Sect. 3. Experimental studies with the different control parameters and comparisons with the different algorithms are presented in the Sect. 4. Finally, in Sect. 5, the conclusion is given.

2 Finding conserved regions in sequences

The evolutionary processes such as mutation, selection and adaptation have lead to considerable divergence between genes of different species that originate from a single gene in the last common ancestor of the species (Mathe 2002; Wang 2004). However, if a nucleotide sequence has a potential to encode a polypeptide or play a vital role to regulate the expression and function of the cell cycle, the conservation or protection of this mentioned region is more likely compared to the other nucleotide sequences and enormous amount of information is conserved throughout the evolution (Mathe 2002; Wang 2004). Conserved regions between sequences that have an evolutionary relationship are the main source of the computational techniques used to identify functional regions including transcription-factor binding sites (Mathe 2002; Wang 2004).

Given a set of N different sequences, each of which contains equal number of sequences, L, and suppose that we try to find N different l nucleotide long sequence segments or motifs, one from each sequence, for which conservation or similarity in terms of nucleotide types on the same locations is higher than or equal to the conservation level or similarity values belonging to other possible group of motifs. To describe l nucleotide long motifs, a start position vector denoted by $p = (p_1, p_2, \ldots, p_N)$ where $1 \le p_i \le (L - l + 1)$ and $1 \le i \le N$ can be defined. By using the start positions stored in the vector p, the nucleotides included in each motifs can be organized into a matrix form that is also called alignment matrix in which ith row corresponds to the motif in the ith complete sequence and jth column of the ith row corresponds to the $(p_i + j - 1)$th nucleotide of the ith sequence in the given set of sequences.

In order to make a discrimination between possible alignment matrices, some statistical information, that can also be utilized to measure the conservation level or inner similarity value, should be used on behalf of these matrices. When an alignment matrix is reduced into a more compact form on the basis of the nucleotide frequencies found in each alignment position, a profile or profile matrix can be obtained (Jones and Pevzner 2004; Zvelebil and Baum 2007; Lesk 2013). In a profile matrix, there are four rows, each of which represents one of the residues in type adenine (A), guanine (G), cytosine (C), thymine (T) and each of the l different columns stores the values that show the presence frequency of the related residue type in that column. A final step to obtain a more genuine representation depends on selecting most frequent residue or residues by utilizing the alignment and profile matrices to form a single nucleotide sequence called consensus string (Jones and Pevzner 2004; Lesk 2013). Generation of a profile matrix from the information stored in the alignment matrix and their further summarization to a consensus string are illustrated in the Fig. 1 for 5 nucleotide length motifs.

One of the commonly used measure to distinguish an alignment matrix from another is based on the simple summation of the highest residue frequencies in each alignment column of a set of N different sequence and given in the Eq. 1 below;

$$Sim(P) = \sum_{i=1}^{l} \frac{max(p(i))}{N \times l} \tag{1}$$

where $max(p(i))$ and l show the maximum frequency of the ith column in the profile matrix and length of the motif, respectively (Kaya 2009; González-Álvarez 2010; Jones and Pevzner 2004; Lesk 2013). Another important approach used in similarity value calculation is finding the entropy of the profile matrix. Given that P is a $4xl$ profile matrix and $p(i, j)$ is the element of the ith row and jth column of the P, the entropy can be calculated as in the Eq. 2 (Kaya 2009; González-Álvarez 2010; Jones and Pevzner 2004; Lesk 2013).

$$Sim(P) = \sum_{j=1}^{l} \sum_{i=1}^{4} \frac{p(i,j)}{N} \log \frac{p(i,j)}{N} \tag{2}$$

3 ABC algorithm and its adaptation to motif discovery problem

In nature, some species especially those living as colonies or swarms demonstrate complex and collective behavior without requiring a centralized monitoring or control mechanism. One of these intelligent behaviors can be seen in the foraging habits of the real honey bee colonies (Karaboga and Basturk 2007, 2008; Akay and Karaboga 2010). In the foragers of a honey bee colony, there are two types of bees; employed and unemployed bees (Karaboga and Aslan 2015; Celik 2016; Badem et al. 2017). Employed bees are responsible to exploit the food sources already discovered and share their information about the location and nectar quality of these food sources with the unemployed bees on the dance area. The information shared by an employed bee is directly proportional to the profitability of the memorized food source (Karaboga and Aslan 2015; Celik 2016; Badem et al. 2017). Unemployed foragers are also classified into two groups called onlooker and scout bees (Karaboga and Aslan 2015; Celik 2016; Badem et al. 2017). While onlooker bees choose a food source by utilizing the information given by the employed bees, scout bees leave the hive to discover new and unvisited rich food sources. If a food source is not worth exploiting anymore, the employed bee associated with that food source becomes a scout bee and joins in the search to find a new food source randomly (Karaboga and Aslan 2015; Celik 2016; Badem et al. 2017).

ABC algorithm which is a population based optimization technique inspired by the mentioned foraging behavior of honey bees was first proposed by Karaboga in 2005 and it has been further developed and applied to various problems since that date (Bolaji 2013; Akay and Karaboga 2015). In ABC algorithm, food sources in the search space correspond to the possible solutions of the problem and nectar amount of these food sources represent the fitness value of the solutions. The whole colony in the ABC algorithm contains equal number of employed and

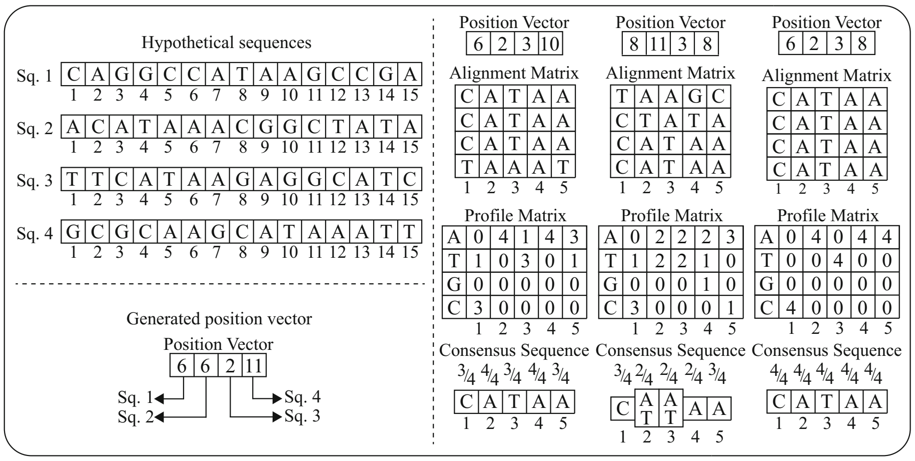

Hypothetical sequences
Sq. 1 C A G G C C A T A A G C C G A
Sq. 2 A C A T A A A C G G C T A T A
Sq. 3 T T C A T A A G A G G C A T C
Sq. 4 G C G C A A G C A T A A A T T
Generated position vector
Position Vector 6 6 2 11
Position Vector
Alignment Matrix
Profile Matrix
Consensus Sequence

Fig. 1 Profile matrices and consensus sequences

onlooker bees and each employed bee is associated with only one food source. When an employed bee finds a new rich food source within the neighbourhood of the source in her memory, she changes the source information in her memory with the new one. Onlooker bees choose food sources based on the information provided by the employed bees and try to find new rich food sources by modifying the position of the chosen food source. The choice of a food source by an onlooker bee is managed with a probabilistic selection schema in which more profitable food sources are more likely to be chosen. The decision about a food source whether or not is worth to exploit by an employed bee is operated with a control parameter called *limit*. If a food source is abandoned by considering the *limit* value, its employed bee becomes a scout bee to determine a new food source randomly (Karaboga and Aslan 2015; Celik 2016). The detailed definition about the ABC algorithm can be obtained from several publications (Karaboga and Basturk 2008; Karaboga and Aslan 2015; Celik 2016). The fundamental steps of the ABC algorithm that reflects the cyclical relationship among employed, onlooker and scout bee phases can be summarized in the Algorithm 1.

Algorithm 1 Fundamental steps of the ABC algorithm

```
1: Initialization:
2:      Assign values to $limit$ and $MCN$ parameters.
3:      Generate $SN$ initial food sources and store best solution into the $x_{best}$.
4: Repeat
5:     //Employed bee phase
6:     for $i \leftarrow 1...SN$ do
7:         Generate new solution $x_{new}$ around the $x_i$.
8:         Calculate fitness value of new solution $fit(x_{new})$.
9:         if $fit(x_{new}) > fit(x_i)$ then
10:            Change $x_i$ with $x_{new}$
11:        end if
12:    end for
13:    //Onlooker bee phase
14:    $sentBees \leftarrow 0, currentSource \leftarrow 1$
15:    Find probability values of each food source.
16:    while $sentBees \neq SN$ do
17:        if $p_{currentSource} > rand(0,1)$ then
18:            $sentBees \leftarrow sentBees + 1$.
19:            Generate new solution $x_{new}$ around the $x_{currentSource}$.
20:            Calculate fitness value of new solution $fit(x_{new})$.
21:            if $fit(x_{new}) > fit(x_{currentSource})$ then
22:                Change $x_{currentSource}$ with $x_{new}$
23:            end if
24:        end if
25:        $currentSource = (currentSource + 1) \bmod SN$.
26:    end while
27:    Update $x_{best}$ if more qualified solution is exist.
28:    //Scout bee phase
29:    Determine the abandoned food source using $limit$ values.
30:    Generate a new source for this abandoned food source.
31:    Replace abandoned source with the newly produced one.
32: Until Maximum cycle number $(MCN)$ is reached
33: Termination:
34:     Print $x_{best}$ as a final solution.
```

3.1 Generating initial food sources

ABC algorithm starts the search process by randomly generating an initial set of food sources that corresponds to the possible solutions. For a numerical optimization problem that needs to optimize D different parameters, jth parameter of the ith food source in the initial set containing SN different food sources is initialized with the Eq. 3;

$$x_{ij} = x_j^{min} + rand(0,1)(x_j^{max} - x_j^{min})\ j = 1,2,\ldots, D \text{ and } i = 1,2,\ldots,SN \tag{3}$$

where x_j^{min} and x_j^{max} are lower and upper bounds of the jth parameter value, respectively (Karaboga and Basturk 2008; Karaboga and Aslan 2015; Celik 2016). When solving motif discovery problem with the ABC algorithm, a vector containing start position of the motifs for each sequence represents a food source, in other words a possible solution. While the lower bound of the start points is clearly equal to 1, the upper bounds of the parameters are determined by using the total number of nucleotides in a sequence and length of motif defined as $(L - l + 1)$.

3.2 Sending employed and onlooker bees to food sources

In the ABC algorithm, each food source is associated with only one bee at the same time and this bee attempts to determine a new food source depending on the location information in its memory. If the nectar quality of the new food source is better than the known source, the bee will forget the previous food source information in her mind and memorize the new source information, which is called as greedy selection mechanism. The mathematical expression used by both employed and onlooker bees to produce a new food source in the neighborhood of the present food source is given in Eq. 4 (Karaboga and Basturk 2008; Karaboga and Aslan 2015; Celik 2016; Karaboga and Aslan 2016b).

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{4}$$

$\phi_i j$ is a random number between $[-1,1]$, $k \in \{1,2,\ldots,SN\}$ and $j \in \{1,2,\ldots,D\}$ are randomly selected indexes. Although, the value of k is randomly determined, it should be noticed that the same values must not assigned to k and i indexes. v_{ij} is the newly produced jth parameter for the solution vector v_i whose parameters have the same value with solution vector x_i except the randomly selected jth parameter value. The main motivation of the mathematical approximation used to generate neighbour solution is to change a parameter by utilizing the information of the randomly selected neighbour food source. To check the validity of this approach on the generation of candidate start position vectors, we first employed all of the terms in the Eq. 4. When an employed or onlooker bee is sent to a new food source around the ith memorized food source, which also corresponds to the position vector and its alignment and profile matrix counterparts, jth parameter value of the ith position vector is updated by adding an integer value, that is calculated by multiplying the difference between the jth parameters of the ith and kth position vectors with the random number defined as ϕ_{ij}, to this parameter. In order to understand the fundamental properties of the first candidate generation model, the motif discovery technique using this ABC model is called ABC_{MOD1}, and the creation of candidate food source around the neighborhood of the ith food source by means of the kth food source source is examined in Fig. 2.

The second new solution generation model depends on the direct usage of the information taken from the neighbour food source instead of using it to change current parameter value of the solution. The fundamental idea lying behind this type of information usage is that if two alignment matrices produce similar consensus sequences, it is possible to find start positions that can be directly usable on the new solution. By considering the fact that a more appropriate subsequence and its start position have already

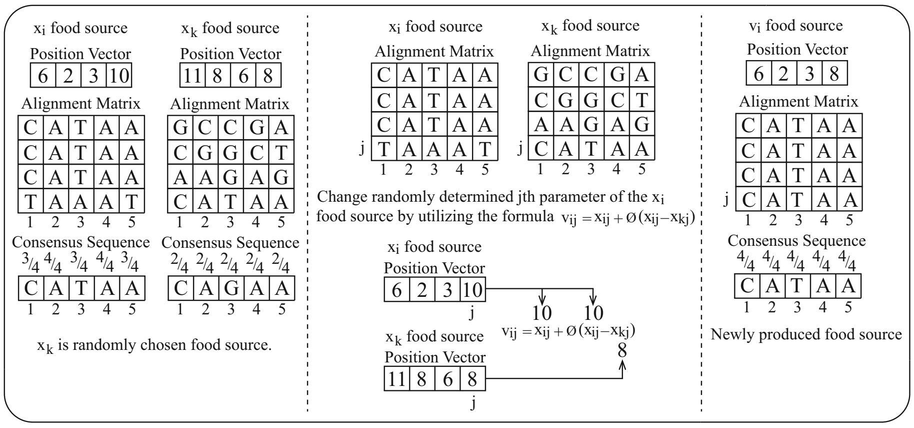
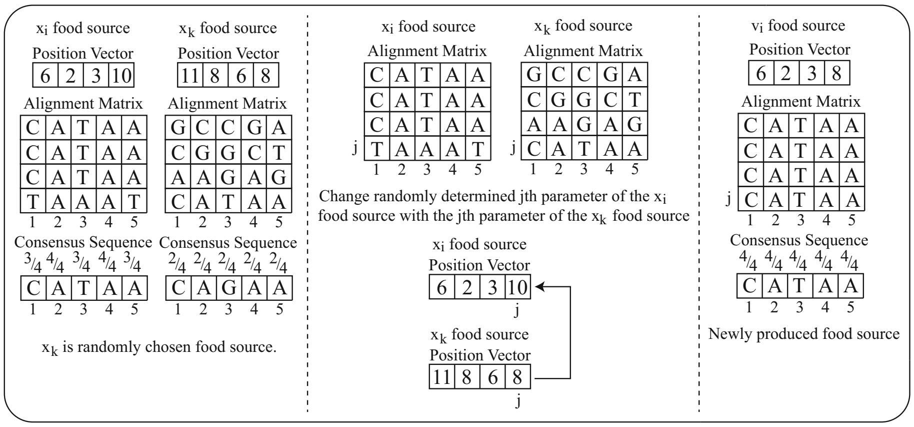


Fig. 2 Generation of new food source by employed or onlooker bees in ABC_{MOD1}



x_i food source
x_k food source
Position Vector
6 2 3 10
11 8 6 8
Alignment Matrix
Consensus Sequence
C A T A A
C A G A A
x_k is randomly chosen food source.
Change randomly determined jth parameter of the x_i food source with the jth parameter of the x_k food source
v_i food source
6 2 3 8
Newly produced food source

Fig. 3 Generation of new food source by employed or onlooker bees in ABC_{MOD2}

been found by the utilized position vector, substitution of an element in a position vector with the corresponding element in the another position vector can lead to increase the similarity value. When an employed or onlooker bee is sent to a new food source by referencing the *i*th memorized food source, *j*th parameter of the *i*th position vector is altered by replacing it with the *j*th parameter of the *k*th position vector. In order to understand the second new solution creation model, the motif discovery technique using this alternative model is called ABC_{MOD2}, production of a food source around the neighborhood of the *i*th food source using the *k*th food source is illustrated in Fig. 3.

3.3 Choosing or abandoning a food source

ABC algorithm accommodates the preference of a food source by an onlooker bee with the nectar amount of that food source. After employed bees share the information about the food sources in their minds on the dance area, an onlooker bee waiting on the hive chooses a food source depending on the probability value associated with the nectar amount of that food source. The selection probability of a food source, which increases with the nectar quality of that source, is calculated by Eq. 5

$$p(x_i) = \frac{fit_i}{\sum_j^{SN} fit_j} \tag{5}$$

where $p(xi)$ is the selection probability and $fit(xi)$ is the fitness value of the solution represented by *i*th index (Karaboga and Basturk 2008; Karaboga and Aslan 2015; Celik 2016). The calculation of the $fit(x_i)$ can be defined based on the minimization or maximization goal of the objective function value. Given that $obj(x_i)$ is the objective function value of the x_i solution, computation of the fitness value can be expressed as in the Eq. 6 (Karaboga and Basturk 2008; Karaboga and Aslan 2015; Celik 2016).

$$fit(x_i) = \begin{Bmatrix} 1 + obj(x_i); & if\ obj(x_i) > 0 \\ 1/(1 + |obj(x_i)|); & if\ obj(x_i) \leq 0 \end{Bmatrix} \tag{6}$$

When the characteristics of the employed and onlooker bee phases are investigated, it is seen that the exploitation process is carried out by both these bee phases. However, in a robust search, a good balance between exploitation and exploration should be maintained. If food source cannot be improved through a predetermined number of iterations or cycles, the employed bee associated with that food source becomes a scout and leave her food source to start a random search process. Scout bees carry out explorations process efficiently. The number of cycles used to determine which food source should be abandoned is an important control parameter of the ABC algorithm called *limit* (Karaboga and Basturk 2008; Karaboga and Aslan 2015; Celik 2016). In the basic ABC algorithm, the food source for which the *limit* value is exceeded the most when compared to the other sources is abandoned and one employed bee becomes a scout (Karaboga and Basturk 2008; Karaboga and Aslan 2015; Celik 2016). For each cycle, only one employed bee can become scout.

4 Experimental studies

To test the performance of the proposed ABC algorithm based motif discovery techniques, named ABC_{MOD1} and ABC_{MOD2}, they were applied to three different data sets extracted from the TRANSFAC database (Matys 2003). The first data set identified with the code *hm03r* for which *hm* abbreviation comes from the human contains 10 sequences whose lengths are equal to 1500 (Matys 2003). The second and third data sets identified with the *yst04r* and *yst08r* for which *yst* abbreviation comes from the yeast contain 7 and 11 sequences, respectively. The lengths of the sequences in *yst04r* and *yst08r* are equal to 1000 (Matys 2003).

Table 1 Results for the *hm03r*, *yst04r* and *yst08r* data sets with the ABC_{MOD1}

Data set	Len.	$limit = 50$			$limit = \infty$		
		Mean	Best	Std.	Mean	Best	Std.
hm03r	9	0.76925	0.80000	1.83813e−02	**0.85259**	0.87777	1.39738e−02
	10	0.75466	0.79000	1.67606e−02	**0.83466**	0.86000	1.25212e−02
	11	0.73484	0.76363	1.39376e−02	**0.81575**	0.86363	1.45088e−02
	12	0.73250	0.78333	1.89966e−02	**0.81416**	0.83333	9.06296e−03
	13	0.71718	0.74615	1.43815e−02	**0.80000**	0.82307	1.12474e−02
	14	0.70547	0.72857	1.10472e−02	**0.78642**	0.80000	8.46193e−03
	16	0.69166	0.71875	1.06791e−02	**0.76729**	0.78750	8.47844e−03
	18	0.67555	0.70000	1.25163e−02	**0.74740**	0.76111	9.31150e−03
	19	0.67263	0.70526	1.10919e−02	**0.73964**	0.75789	7.01137e−03
	21	0.65841	0.68095	1.03071e−02	**0.72492**	0.74761	9.66081e−03
yst04r	8	0.90059	0.94642	1.60291e−02	**0.95476**	0.96428	1.02026e−02
	9	0.88148	0.90476	1.30048e−02	**0.92486**	0.95238	1.01537e−02
	13	0.83003	0.85714	1.28196e−02	**0.88131**	0.90109	1.27108e−02
	15	0.80666	0.81904	8.35166e−03	**0.85079**	0.86666	6.29436e−03
	16	0.79494	0.82142	9.51965e−03	**0.84375**	0.84821	6.52739e−03
	17	0.78431	0.79831	8.35484e−03	**0.82745**	0.84033	5.72663e−03
	20	0.77095	0.78571	8.98277e−03	**0.81071**	0.81428	4.08830e−03
	22	0.75779	0.77922	7.84244e−03	**0.80800**	0.81168	6.53426e−03
yst08r	10	0.81666	0.85454	1.21928e−02	**0.90424**	0.91818	6.63912e−03
	11	0.79779	0.81818	1.22260e−02	**0.89008**	0.90082	5.80309e−03
	14	0.76926	0.79220	1.43419e−02	**0.83787**	0.85064	7.52702e−03
	16	0.75662	0.77272	9.44680e−03	**0.81534**	0.82386	4.41388e−03
	17	0.74902	0.75935	9.51766e−03	**0.80695**	0.81283	4.29417e−03
	21	0.73232	0.75757	1.06696e−02	**0.78412**	0.78787	3.35964e−03

Bold values show the best results of the given algorithms

In the first part of the experimental studies, serial implementations of the ABC_{MOD1} and ABC_{MOD2} algorithms were carried out with the different values of the *limit* parameter and their characteristics were analyzed in terms of similarity value of the predicted motifs and convergence performance. In the second part of the experimental studies, shared memory based parallelization of the ABC_{MOD1} and ABC_{MOD2} algorithms, $PABC_{MOD1}$ and $PABC_{MOD2}$ algorithms, were described and tested with the different values of the *limit* parameter by utilizing the ring migration topology. Finally, the results obtained by using both sequential and parallel implementations of the ABC_{MOD1} and ABC_{MOD2} algorithms were compared with the results of the GA based technique called MOGAMOD (Kaya 2009), DE algorithm based technique called DEPT (González-Álvarez 2010) and previously proposed ABC algorithm based motif discovery techniques called MOABC (González-Álvarez 2011) and consensusABC (Karaboga and Aslan 2016a) in the Sect. 4.3.[1]

[1] Please send an e-mail to the corresponding author for the results of the all experimental cases.

4.1 Serial implementations of ABC_{MOD1} and ABC_{MOD2} algorithms

The first stage of the experimental studies was devoted to the investigation of the serial implementations of the ABC_{MOD1} and ABC_{MOD2} algorithms. Serial implementations of the ABC_{MOD1} and ABC_{MOD2} were used to identify 9, 10, 11, 12, 13, 14, 16, 18, 19 and 21 nucleotide length transcription factor binding sites on *hm03r* data set, 8, 9, 13, 15, 16, 17, 20 and 22 nucleotide length transcription factor binding sites on *yst04r* data set and 10, 11, 14, 16, 17 and 21 nucleotide length transcription factor binding sites on *yst08r* data set. To make a fair comparison with the other evolutionary computation techniques, number of food sources was taken equal to 100 and the maximum number of cycle was set to 3000 (Kaya 2009; González-Álvarez 2010, 2011; Karaboga and Aslan 2016a). For the motifs being identified with ABC_{MOD1} and ABC_{MOD2}, all of the sequences in a data set were taken into account on the calculation of the similarity value given in the Eq. 1 (Kaya 2009; González-Álvarez 2010, 2011; Karaboga and Aslan 2016a). In other words, with respect to the decreasing in

Table 2 Results for the *hm03r*, *yst04r* and *yst08r* data sets with the ABC_{MOD2}

Data set	Len.	*limit* = 50			*limit* = ∞		
		Mean	Best	Std.	Mean	Best	Std.
hm03r	9	**0.86925**	0.90000	1.38972e-02	0.74888	0.77777	1.53297e–02
	10	**0.84966**	0.87000	1.09806e-02	0.73666	0.78000	1.78756e–02
	11	**0.83242**	0.86363	1.76542e-02	0.72151	0.76363	1.59228e–02
	12	**0.82222**	0.85833	1.44060e-02	0.71444	0.76666	1.99296e–02
	13	**0.81153**	0.83846	1.79266e-02	0.69717	0.73076	1.76667e–02
	14	**0.79166**	0.81428	1.09511e-02	0.69095	0.72142	1.59094e–02
	16	**0.76812**	0.78750	1.53400e-02	0.67833	0.70000	1.41890e–02
	18	**0.75148**	0.76666	1.48021e-02	0.66000	0.69444	1.55364e–02
	19	**0.74035**	0.75789	1.12571e-02	0.65298	0.67368	1.16410e–02
	21	**0.72587**	0.74761	1.24226e-02	0.65031	0.70476	1.95614e–02
yst04r	8	**0.96071**	0.98214	1.18636e-02	0.84583	0.89285	2.06993e–02
	9	**0.92010**	0.95238	1.41256e-02	0.82751	0.88888	2.42105e–02
	13	**0.86996**	0.89011	1.00315e-02	0.77435	0.82417	2.03515e–02
	15	**0.83873**	0.86666	1.54041e-02	0.75841	0.79047	1.74149e–02
	16	**0.82261**	0.84821	1.27855e-02	0.74583	0.77678	1.63686e–02
	17	**0.81820**	0.84033	1.23824e-02	0.74902	0.78151	1.49241e–02
	20	**0.79452**	0.80714	6.68056e-03	0.72690	0.75714	1.68491e–02
	22	**0.78203**	0.81168	1.27249e-02	0.71450	0.74026	1.37994e–02
yst08r	10	**0.91545**	0.91818	5.41790e-03	0.79393	0.84545	1.58961e–02
	11	**0.88843**	0.90082	6.77667e-03	0.78980	0.83471	1.91290e–02
	14	**0.82467**	0.83766	6.14835e-03	0.75216	0.79220	1.68885e–02
	16	**0.80814**	0.82386	7.99146e-03	0.73200	0.75568	1.61455e–02
	17	**0.80445**	0.81818	7.11711e-03	0.72477	0.74866	1.29271e–02
	21	**0.77114**	0.78787	1.22301e-02	0.70562	0.73593	1.45143e–02

Bold values show the best results of the given algorithms

the similarity value, all nucleotide sequences in the data set were used as a candidate in which at least one motif is common with the remainder of the data set (Kaya 2009; González-Álvarez 2010, 2011). For each motif length, 30 independent runs were carried out by ABC_{MOD1} and ABC_{MOD2} algorithms with different values of *limit* parameter, including 50, 100, 200, 300, 500 and ∞ and the best, mean best similarity values and standard deviations were recorded. When the recorded results were evaluated, it has seen that similarity values of the obtained motifs by the ABC_{MOD1} algorithm increase with the ascending values of the *limit* parameter and the highest similarity values are mostly obtained by setting *limit* parameter to ∞ which means that the scout bee phase is not included in the work flow of the algorithm. In the new solution generation schema used for employed and onlooker bee phases of the ABC_{MOD1} algorithm, modifying selected start position of the current position vector by utilizing the known start position and corresponding start position of the randomly chosen position vector contributes to the exploration ability of the bees as well as the exploitation ability. For more clear discrimination how the similarity values change with the *limit* parameters, only the results of the ABC_{MOD1} obtained by setting *limit* parameter value to 50 and ∞ were given in the Table 1.

An analysis about the relationship between *limit* parameter values and similarity values of the obtained motifs can also be made for the ABC_{MOD2} algorithm. When the obtained results for *hm03r*, *yst04r* and *yst08r* data sets were examined, it has seen that the results in terms of best similarity values get better as the value of the *limit* parameter decreases and highest similarity values are mostly obtained by setting *limit* parameter to 50. Although utilization of the selected start position of the randomly determined solution directly on the newly created position vector in employed and onlooker bee phases of the ABC_{MOD2} algorithm increases the probability of the usage from more appropriate parameters, it slightly restricts the

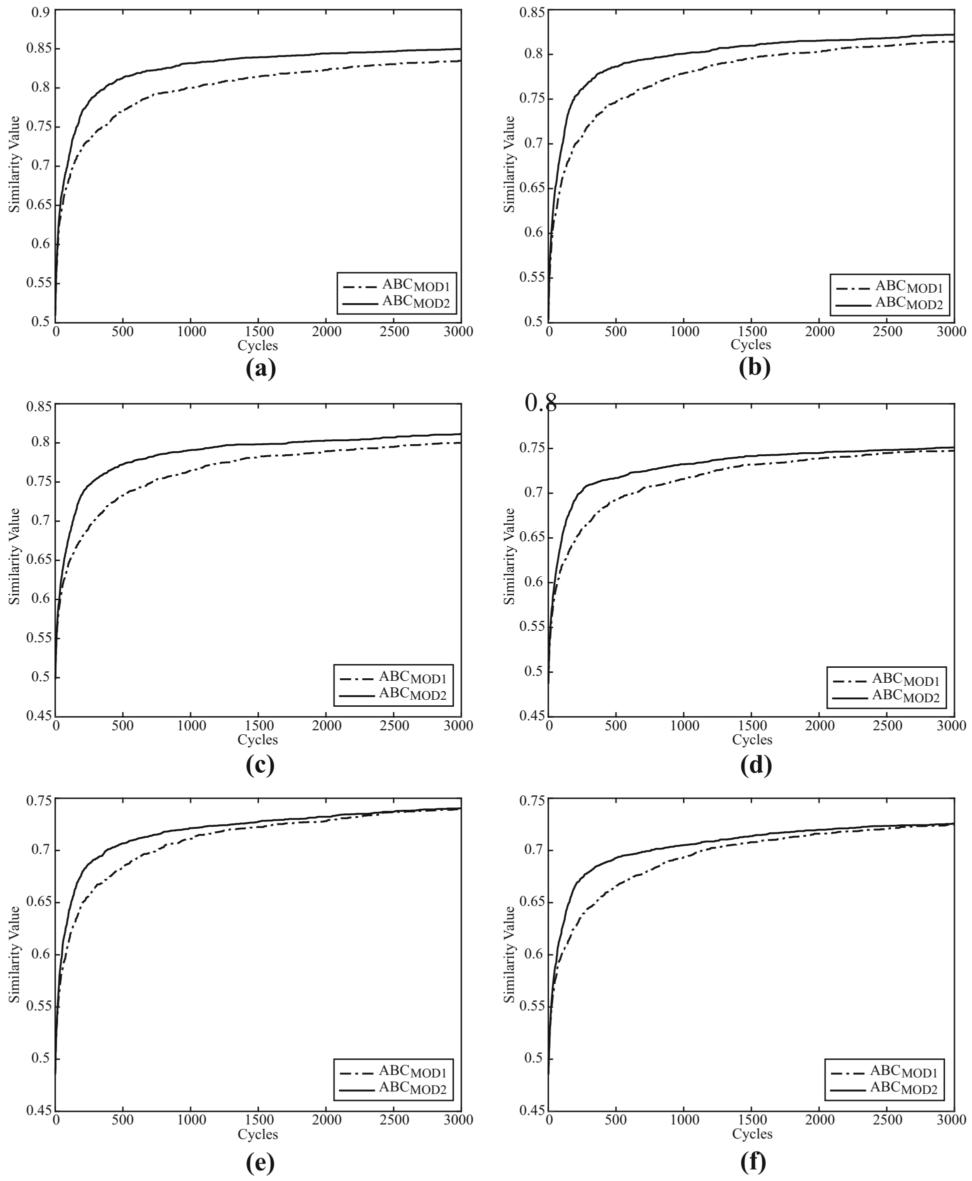


Fig. 4 Mean best similarities of ABC_{MOD1} and ABC_{MOD2} algorithms on 10 (**a**), 12 (**b**), 13 (**c**), 18 (**d**), 19 (**e**) and 21 (**f**) nucleotide length motifs for *hm03r* data set

search ability of the algorithm and the role of the scout bee phase for discovering more qualified motifs becomes more critical when compared to the ABC_{MOD1}. As in the case of ABC_{MOD1} algorithm, only the results of the ABC_{MOD2} obtained by setting *limit* parameter value to 50 and ∞ were given in the Table 2.

Considering the mean best similarity values and maximum cycle numbers, convergence graphics of ABC_{MOD1} when limit parameter value is set to ∞ and ABC_{MOD2} when *limit* parameter value is set to 50 were given in Figs. 4 and 5 for the detailed investigation of the proposed solution creation approaches for *hm03r* and *yst08r* data sets. On the *hm03r* data set, convergence performance of the ABC_{MOD2} algorithm is significantly better than the convergence performance of the ABC_{MOD1} algorithm during the first half of the total cycles. However, this type of quick convergence is not sustainable until the end of the cycles because of difficulties of the problem and the used model of the ABC_{MOD2} algorithm. When the availability of the motifs in sequences, length of the motifs being discovered and number of sequences in the data set are taken into account, it should be noted that a straightforward way does not exist for handling all of these exceptional situations. Although the convergence of the ABC_{MOD2} is much quicker than that of the ABC_{MOD1} algorithm, ABC_{MOD1} catches ABC_{MOD2} after the completion of the first 1000 cycles for *yst08r* data set and then continues to improve the mean best similarity values.

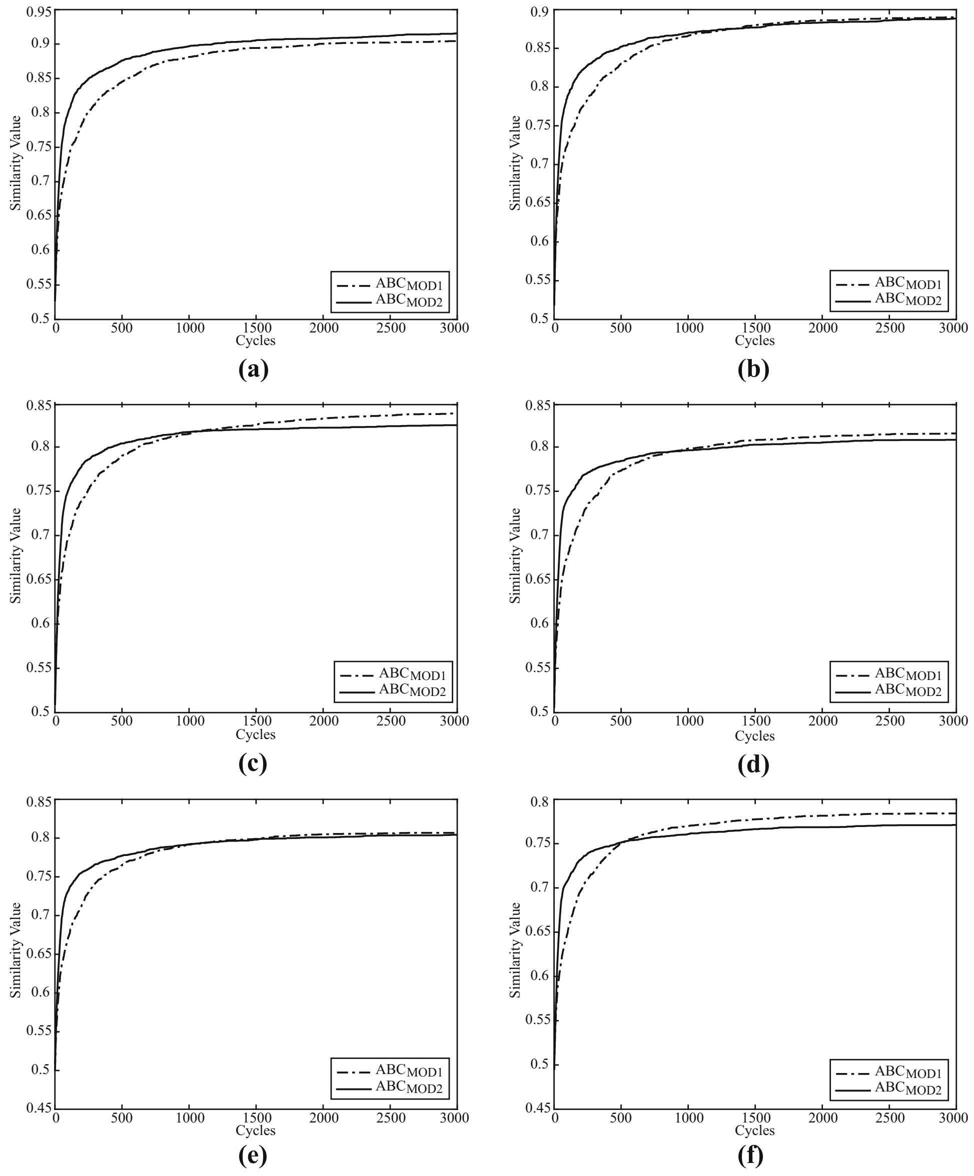


Fig. 5 Mean best similarities of ABC_{MOD1} and ABC_{MOD2} algorithms on 10 (**a**), 11 (**b**), 14 (**c**), 16 (**d**), 17 (**e**) and 21 (**f**) nucleotide length motifs for *yst08r* data set

4.2 Parallel implementation of the ABC_{MOD1} and ABC_{MOD2} algorithms

Dividing the whole population or solutions into subpopulations and then assigning them to the different computation nodes or processes working simultaneously are probably the most preferred type of parallel programming model used to increase the overall running performance of the evolutionary computation techniques. In this parallelization model, each compute node is in communication by changing selected individual with the other subcolonies based on a neighborhood topology and migration period that controls the communication frequency between subcolonies. Like other population based algorithms, ABC algorithm is intrinsically suitable to run in parallel on shared or distributed memory architectures. However, some stages of the ABC algorithm require synchronized workflow especially for the sequential sending of the bees and the calculation of the probabilities. Because of these reasons, if a modification is made on the mechanism of the algorithm or it is used to solve a combinatorial problem, its performance on the multi-core or multi-processor systems should be analyzed.

In order to further investigate the performance of the ABC algorithm models, we parallelized ABC_{MOD1} and ABC_{MOD2} with the ring migration topology in which the best individual in the *i*th subpopulation is transferred to the $((i \; mod \; N) + 1)$th subpopulation, where N shows the number of subpopulations, to change it with the worst individual in the $((i \; mod \; N) + 1)$th subpopulation. The migration interval was set to 20 which means that after completing each successive 20 cycles, subpopulations send

Table 3 Results for the *hm03r*, *yst04r* and *yst08r* data sets with the parallel ABC_{MOD1}

Data set	Len.	*limit* = 50			*limit* = 500		
		Mean	Best	Std.	Mean	Best	Std.
hm03r	9	0.78592	0.83333	1.67452e–02	**0.87185**	0.88888	9.10340e–03
	10	0.76833	0.81000	1.66263e–02	**0.84900**	0.88000	1.02889e–02
	11	0.74666	0.79090	1.48582e–02	**0.83818**	0.85454	1.15486e–02
	12	0.74027	0.76666	1.23958e–02	**0.83222**	0.85833	1.19331e–02
	13	0.73128	0.75384	1.27655e–02	**0.81974**	0.83846	1.08190e–02
	14	0.72047	0.74285	1.29598e–02	**0.80571**	0.82142	6.86658e–03
	16	0.70687	0.73125	1.32684e–02	**0.78229**	0.79375	7.88579e–03
	18	0.69018	0.71666	1.02837e–02	**0.76222**	0.77777	6.74916e–03
	19	0.68280	0.71052	1.13079e–02	**0.75245**	0.76842	6.97734e–03
	21	0.67238	0.70000	1.14203e–02	**0.74095**	0.76190	8.62781e–03
yst04r	8	0.90119	0.94642	1.53637e–02	**0.96309**	0.98214	6.52050e–03
	9	0.88677	0.90476	1.08165e–02	**0.93280**	0.95238	1.36247e–02
	13	0.83736	0.86813	1.33500e–02	**0.88168**	0.90109	1.46251e–02
	15	0.80952	0.84761	1.17310e–02	**0.85079**	0.86666	6.77298e–03
	16	0.79881	0.82142	1.23480e–02	**0.84404**	0.84821	7.31526e–03
	17	0.79187	0.80672	9.01385e–03	**0.83193**	0.84033	7.64467e–03
	20	0.77547	0.79285	9.31943e–03	**0.81404**	0.81428	1.30409e–03
	22	0.76255	0.78571	7.56562e–03	**0.81082**	0.81168	3.71003e–03
yst08r	10	0.83757	0.87272	1.34487e–02	**0.91242**	0.91818	7.72987e–03
	11	0.82341	0.86776	1.47971e–02	**0.89283**	0.90082	5.93714e–03
	14	0.78008	0.79870	8.81701e–03	**0.84480**	0.85064	4.92767e–03
	16	0.76931	0.79545	9.02701e–03	**0.81856**	0.82386	4.20271e–03
	17	0.76345	0.78609	9.17495e–03	**0.81016**	0.81818	4.38507e–03
	21	0.74805	0.76190	7.40267e–03	**0.78744**	0.79220	1.74276e–03

Bold values show the best results of the given algorithms

and receive the best individuals based on the ring neighborhood topology. Each of the experiments was repeated 30 times with different random seeds for 100 food sources divided into 4 subpopulations and executed on the system that is equipped with Intel® i7 2600 processor at 3.40 GHz and 4 gigabytes of main memory running Fedora 22. Serial and parallel implementations of the ABC_{MOD1} and ABC_{MOD2} algorithms were coded in C programming language and *Pthreads* library was used for satisfying the required shared memory based parallelism.

Best and average similarity values of the predicted motifs with different lengths over 3000 cycles for *limit* values including 50, 100, 200, 300, 500 and ∞ as in the serial implementations were recorded. From the recorded results, it has found that the effect of the *limit* parameter on the parallel ABC_{MOD1} algorithm performance is different from the serial implementation. For serial implementation of the ABC_{MOD1}, the best similarity values were obtained by setting the *limit* parameter to ∞, while the best similarity values were obtained by setting the limit parameter to 200, 300, 500 or ∞ for mentioned parallel implementation of the algorithm. In the case of *hm03r* data set, while the highest similarity values were found by setting the *limit* parameter to 500 for 9, 10, 12, 13, 16, 18, 19 and 21 nucleotide length motifs, highest similarity values for 11 and 14 nucleotide length motifs were found by setting *limit* parameter as ∞ and 300, respectively. For the *yst04r* data set, ABC_{MOD1} algorithm was capable of finding highest similarity values when the value of the *limit* parameter was equal to 300 or 500. Finally, in the case of *yst08r* data set, the parallelized algorithm achieved the best results when the *limit* value was 500. The results of the parallel ABC_{MOD1} obtained by setting *limit* parameter value to 50 and 500 were given in Table 3.

Parallelization of the ABC_{MOD2} also slightly changes its search characteristics when compared to the sequential counterpart of the algorithm. In parallel ABC_{MOD2} algorithm, the discovery of the highly conserved motifs can be done with different values of the *limit* parameter between 50 and 200 as in the sequential ABC_{MOD2}. In the case of *hm03r* data set, while the highest similarity values were found by setting the *limit* parameter to 50 for 9, 10, 12, 13,

Table 4 Results for the *hm03r*, *yst04r* and *yst08r* data sets with the parallel ABC_{MOD2}

Data set	Len.	*limit* = 50			*limit* = ∞		
		Mean	Best	Std.	Mean	Best	Std.
hm03r	9	**0.88111**	0.90000	8.33013e−03	0.73888	0.78888	2.71774e−02
	10	**0.85233**	0.87000	1.35655e−02	0.72700	0.77000	2.54815e−02
	11	**0.84363**	0.86363	1.35892e−02	0.71272	0.76363	2.23447e−02
	12	**0.83305**	0.85833	1.20827e−02	0.69833	0.73333	2.32015e−02
	13	**0.81769**	0.84615	1.44475e−02	0.69461	0.72307	1.56671e−02
	14	**0.80142**	0.82857	1.43349e−02	0.68738	0.72857	2.21697e−02
	16	**0.77979**	0.80625	1.25734e−02	0.66895	0.70625	2.02012e−02
	18	**0.75555**	0.77222	9.89515e−03	0.64592	0.69444	1.63724e−02
	19	**0.74789**	0.76315	1.00004e−02	0.64807	0.70000	1.81129e−02
	21	**0.73254**	0.75238	9.77367e−03	0.63714	0.67619	1.22555e−02
yst04r	8	**0.96190**	0.98214	9.06100e−03	0.82500	0.87500	2.31645e−02
	9	**0.92592**	0.95238	1.33997e−02	0.81851	0.85714	1.98531e−02
	13	**0.87802**	0.90109	1.66656e−02	0.77545	0.81318	1.88540e−02
	15	**0.84730**	0.86666	1.21201e−02	0.75206	0.78095	1.55154e−02
	16	**0.83392**	0.84821	1.18636e−02	0.74315	0.78571	1.68567e−02
	17	**0.82605**	0.84033	1.08449e−02	0.74453	0.77310	1.73204e−02
	20	**0.80285**	0.81428	7.64262e−03	0.72333	0.75000	1.52314e−02
	22	**0.78744**	0.81168	1.10426e−02	0.71363	0.74675	1.79945e−02
yst08r	10	**0.91697**	0.91818	3.94701e−03	0.79545	0.83636	2.45506e−02
	11	**0.88567**	0.90082	6.17057e−03	0.78347	0.81818	2.11984e−02
	14	**0.83679**	0.85064	1.25002e−02	0.74199	0.77277	1.60815e−02
	16	**0.81572**	0.82386	6.44954e−03	0.72878	0.76136	1.64087e−02
	17	**0.80855**	0.81818	6.18551e−03	0.71907	0.74866	1.66599e−02
	21	**0.78095**	0.78787	6.07976e−03	0.70418	0.73593	1.34193e−02

Bold values show the best results of the given algorithms

16, 19 and 21 nucleotide length motifs, highest similarity value for 18 nucleotide length motif was found by setting *limit* parameter as 100. For *yst04r* data set, parallel ABC_{MOD2} algorithm showed almost the same performance on detecting best similarity values when the *limit* parameter was chosen as 50 or 100. Finally, in the case of *yst08r* data set, it was seen that ABC_{MOD2} algorithm was capable of finding highest similarity values when the *limit* value was equal or less than 200. The results of the parallel ABC_{MOD2} obtained by setting *limit* parameter value to 50 and ∞ were given in Table 4.

When an algorithm is parallelized for executing on a shared or distributed memory architecture, it should be noticed that the parallelized implementation is not only used to increase the computation speed both also maintains the quality of final solutions and convergence characteristics as in the serial case. In order to compare the convergence performance of the mean best similarity values obtained by the parallel ABC_{MOD1} and ABC_{MOD2} algorithms when *limit* parameter is set to 500 and 50, respectively, with those of their serial counterparts, Figs. 6 and 7 can be analyzed. For all data sets, when the graphics are examined, it is concluded that convergence characteristics of parallel implementations of the ABC_{MOD1} and ABC_{MOD2} are more robust than the convergence characteristics of the serial implementations of them.

In the literature, speedup and efficiency are the most commonly used metrics to measure the performance of the parallel algorithms. Speedup value is the ratio of sequential execution time to parallel execution time and efficiency is the ratio of speedup to the number of process used. Optimum value of the speedup metric is equal to the number of processors and optimum value of the efficiency is equal to 1. By considering the formal definition of the speedup and efficiency metrics, average running times in terms of seconds over 30 independent experiments for serial and parallel implementations of the ABC_{MOD1} and ABC_{MOD2} algorithms, their speedup and efficiency values were

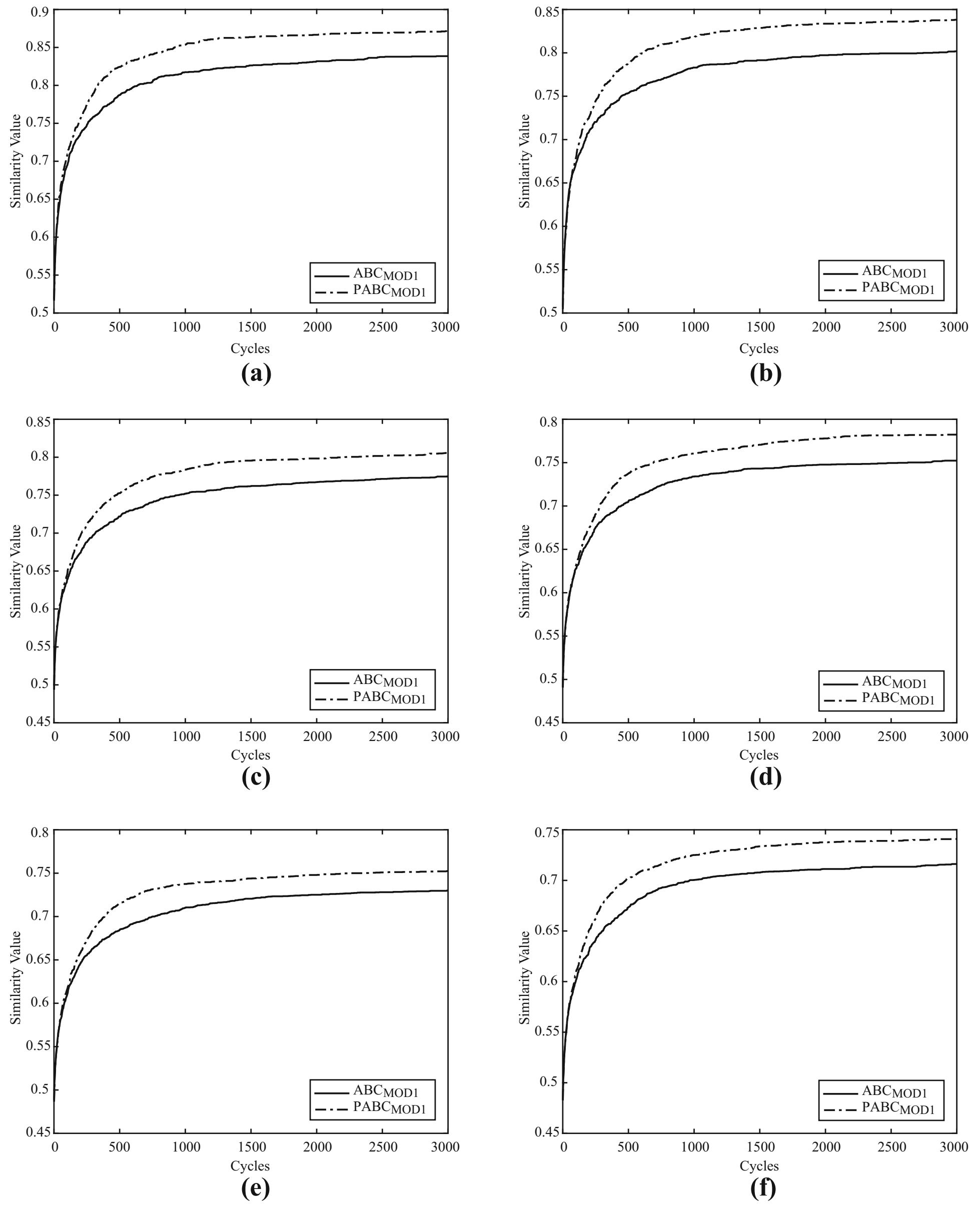


Fig. 6 Mean best similarity values generated by serial and parallel ABC_{MOD1} algorithms on 9 (**a**), 11 (**b**), 14 (**c**), 16 (**d**), 19 (**e**) and 21 (**f**) nucleotide length motifs for *hm03r* data set

summarized in the Tables 5 and 6 for *hm03r* data set. Our observation from these experiments is that there is a relationship between the length of motif being discovered and the speedup and efficiency values of the parallel algorithms. When the motifs are getting longer, speedup and efficiency values of the parallel ABC_{MOD1} algorithm get closer to 3.28 and 0.83. Similar conclusions can be obtained from the results given in Table 6 for parallel ABC_{MOD2} algorithm. When the motifs are getting longer, speedup and efficiency values of the parallel ABC_{MOD2} algorithm get closer to 2.56 and 0.64.

4.3 Performance comparison with other population-based algorithms

In this section, we compare the results obtained by the serial and parallel implementation of the ABC_{MOD1} and ABC_{MOD2} algorithms with other evolutionary and swarm intelligence based motif discovery techniques including MOGAMOD, DEPT, MOABC and consensusABC algorithms in the literature. MOGAMOD, DEPT and MOABC algorithms were multi-objective solving techniques in which number of sequences used to generate consensus sequence, length of motif and its similarity value are tried to be optimized simultaneously (Kaya 2009; González-Álvarez 2010, 2011). Besides, the final pareto fronts of these multiobjective techniques were created by combining

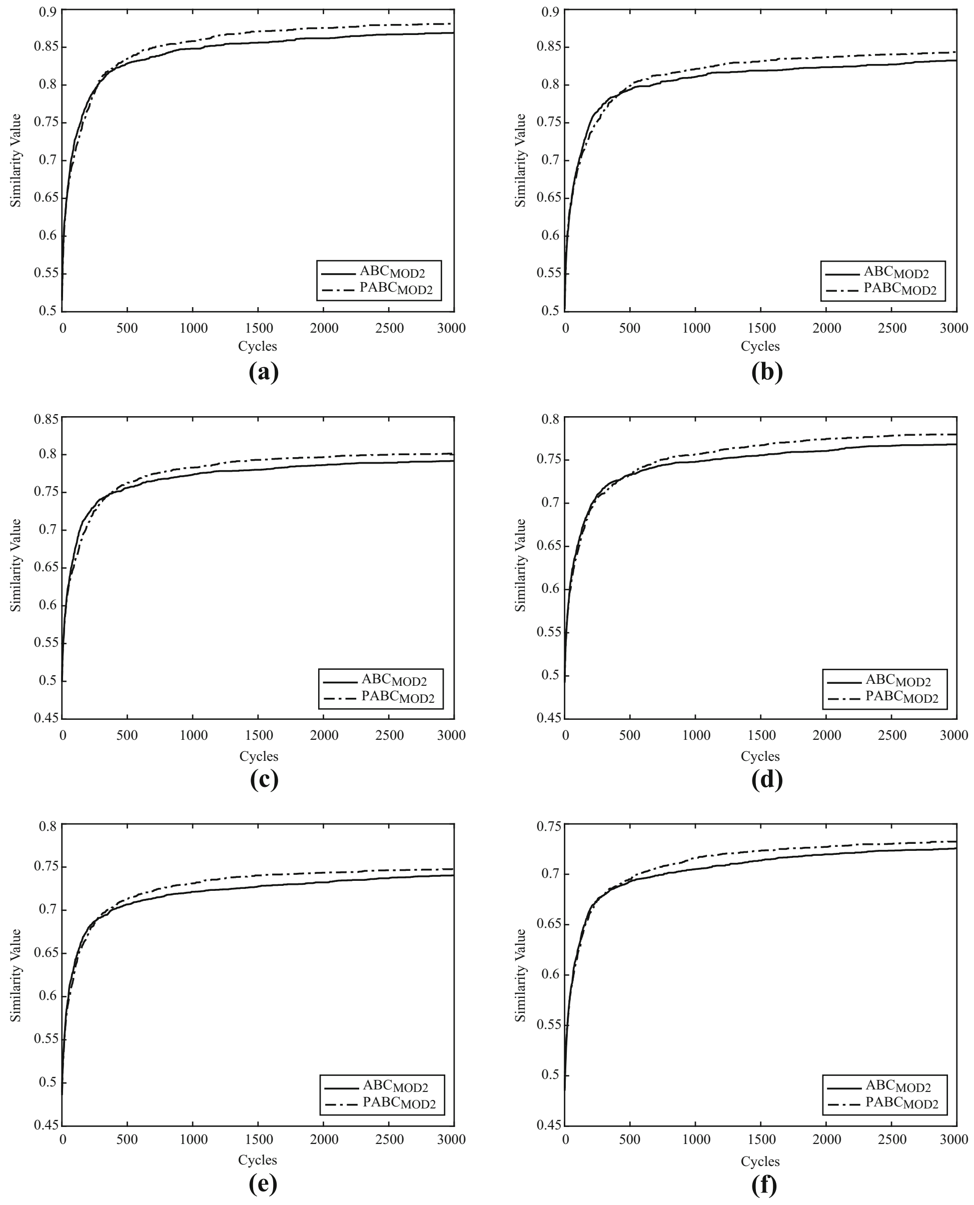


Fig. 7 Mean best similarity values generated by serial and parallel ABC_{MOD2} algorithms on 9 (**a**), 11 (**b**), 14 (**c**), 16 (**d**), 19 (**e**) and 21 (**f**) nucleotide length motifs for *hm03r* data set

Table 5 Speedup and efficiency values of the parallel ABC_{MOD1} on *hm03r* data set

Len.	*limit* = 50				*limit* = ∞			
	ABC_{MOD1}	$PABC_{MOD1}$	Sp.	Ef.	ABC_{MOD1}	$PABC_{MOD1}$	Sp.	Ef.
9	2.196	1.108	1.981	0.495	1.989	0.975	2.039	0.509
10	2.394	1.100	2.175	0.544	2.180	1.001	2.178	0.544
11	2.585	1.126	2.294	0.573	2.314	1.026	2.253	0.563
12	2.754	1.146	2.402	0.600	2.498	1.052	2.373	0.593
13	2.929	1.173	2.495	0.623	2.657	1.072	2.477	0.619
14	3.143	1.234	2.545	0.636	2.857	1.087	2.627	0.657
16	3.418	1.321	2.586	0.646	3.284	1.128	2.909	0.727
18	3.788	1.420	2.666	0.666	3.568	1.172	3.044	0.761
19	3.932	1.434	2.740	0.685	3.867	1.210	3.196	0.799
21	4.282	1.503	2.847	0.711	4.219	1.286	3.279	0.820

Table 6 Speedup and efficiency values of the parallel ABC_{MOD2} on *hm03r* data set

Len.	$limit = 50$				$limit = \infty$			
	ABC_{MOD1}	$PABC_{MOD2}$	Sp.	Ef.	ABC_{MOD2}	$PABC_{MOD2}$	Sp.	Ef.
9	1.837	1.019	1.802	0.450	2.088	0.936	2.230	0.557
10	1.944	0.966	2.013	0.503	2.087	0.952	2.192	0.548
11	2.063	1.000	2.062	0.515	2.072	0.926	2.236	0.559
12	2.204	1.023	2.144	0.536	2.117	0.973	2.175	0.543
13	2.321	1.035	2.242	0.560	2.298	0.993	2.312	0.578
14	2.411	1.078	2.235	0.558	2.287	1.001	2.284	0.571
16	2.541	1.147	2.215	0.553	2.576	1.046	2.461	0.615
18	2.787	1.239	2.248	0.562	2.652	1.084	2.445	0.611
19	2.962	1.213	2.441	0.610	2.771	1.122	2.468	0.617
21	3.189	1.295	2.461	0.615	3.128	1.225	2.553	0.638

Table 7 Comparison of other techniques with ABC_{MOD1} and ABC_{MOD2} on *hm03r*

Len.	MOGAMOD (Kaya 2009)	ABC_{MOD1}	ABC_{MOD2}	$PABC_{MOD1}$	$PABC_{MOD2}$
9	0.81	0.87	**0.90**	0.88	**0.90**
10	0.79	0.86	0.87	**0.88**	0.87
11	0.74	**0.86**	**0.86**	0.85	**0.86**
Len.	DEPT (González-Álvarez 2010)	ABC_{MOD1}	ABC_{MOD2}	$PABC_{MOD1}$	$PABC_{MOD2}$
9	0.85	0.87	**0.90**	0.88	**0.90**
10	0.83	0.86	0.87	**0.88**	0.87
11	0.81	**0.86**	**0.86**	0.85	**0.86**
13	0.81	0.82	0.83	0.83	**0.84**
14	0.79	0.80	0.81	**0.82**	**0.82**
18	0.75	0.76	0.76	**0.77**	**0.77**
19	0.74	0.75	0.75	**0.76**	**0.76**
21	0.74	0.74	0.74	**0.76**	0.75
Len.	MOABC (González-Álvarez 2011)	ABC_{MOD1}	ABC_{MOD2}	$PABC_{MOD1}$	$PABC_{MOD2}$
9	0.84	0.87	**0.90**	0.88	**0.90**
10	0.81	0.86	0.87	**0.88**	0.87
11	0.80	**0.86**	**0.86**	0.85	**0.86**
12	0.79	0.83	**0.85**	**0.85**	**0.85**
Len.	consensusABC (Karaboga and Aslan 2016a)	ABC_{MOD1}	ABC_{MOD2}	$PABC_{MOD1}$	$PABC_{MOD2}$
9	0.86	0.87	**0.90**	0.88	**0.90**
10	0.86	0.86	0.87	**0.88**	0.87
11	0.83	**0.86**	**0.86**	0.85	**0.86**
14	0.80	0.82	0.83	0.83	**0.84**

Bold values show the best results of the given algorithms

pareto fronts found in 30 independent runs (Kaya 2009; González-Álvarez 2010, 2011). In other words, final pareto fronts contained the solutions with the highest similarity values (Kaya 2009; González-Álvarez 2010, 2011). For consensusABC algorithm, each test case was repeated 30 times with different random seeds and all the sequences in the data set were used for discovering the motif whose length is determined in advance (Karaboga and Aslan 2016a).

Making a fair assessment between these mentioned algorithms and proposed models, we kept constant the number of sequences and motif length objectives as in the comparison of the DEPT and MOGAMOD algorithms and only considered the best similarity values (Kaya 2009;

Table 8 Comparison of other techniques with ABC_{MOD1} and ABC_{MOD2} on *yst04rs*

Len.	MOGAMOD (Kaya 2009)	ABC_{MOD1}	ABC_{MOD2}	$PABC_{MOD1}$	$PABC_{MOD2}$
8	0.84	0.96	**0.98**	**0.98**	**0.98**
9	0.80	**0.95**	**0.95**	**0.95**	**0.95**
Len.	DEPT (González-Álvarez 2010)	ABC_{MOD1}	ABC_{MOD2}	$PABC_{MOD1}$	$PABC_{MOD2}$
8	0.96	0.96	**0.98**	**0.98**	**0.98**
9	0.92	**0.95**	**0.95**	**0.95**	**0.95**
15	**0.86**	**0.86**	**0.86**	**0.86**	**0.86**
16	**0.84**	**0.84**	**0.84**	**0.84**	**0.84**
17	**0.84**	**0.84**	**0.84**	**0.84**	**0.84**
22	0.80	**0.81**	**0.81**	**0.81**	**0.81**
Len.	MOABC (González-Álvarez 2011)	ABC_{MOD1}	ABC_{MOD2}	$PABC_{MOD1}$	$PABC_{MOD2}$
8	0.94	0.96	**0.98**	**0.98**	**0.98**
9	0.90	**0.95**	**0.95**	**0.95**	**0.95**
13	0.84	**0.90**	0.89	**0.90**	**0.90**
20	0.80	0.81	0.80	0.81	0.81
Len.	consensusABC (Karaboga and Aslan 2016a)	ABC_{MOD1}	ABC_{MOD2}	$PABC_{MOD1}$	$PABC_{MOD2}$
8	0.96	0.96	**0.98**	**0.98**	**0.98**
9	0.93	**0.95**	**0.95**	**0.95**	**0.95**
13	0.87	**0.90**	0.89	**0.90**	**0.90**
22	0.80	**0.81**	**0.81**	**0.81**	**0.81**

Bold values show the best results of the given algorithms

Table 9 Comparison of other techniques with ABC_{MOD1} and ABC_{MOD2} on *yst08r*

Len.	MOABC (González-Álvarez 2011)	ABC_{MOD1}	ABC_{MOD2}	$PABC_{MOD1}$	$PABC_{MOD2}$
11	0.85	**0.90**	**0.90**	**0.90**	**0.90**
14	0.80	**0.85**	0.83	**0.85**	**0.85**
Len.	MOGAMOD (Kaya 2009)	ABC_{MOD1}	ABC_{MOD2}	$PABC_{MOD1}$	$PABC_{MOD2}$
10	0.80	**0.91**	**0.91**	**0.91**	**0.91**
11	0.77	**0.90**	**0.90**	**0.90**	**0.90**
Len.	DEPT (González-Álvarez 2010)	ABC_{MOD1}	ABC_{MOD2}	$PABC_{MOD1}$	$PABC_{MOD2}$
10	0.89	**0.91**	**0.91**	**0.91**	**0.91**
11	0.88	**0.90**	**0.90**	**0.90**	**0.90**
16	**0.82**	**0.82**	**0.82**	**0.82**	**0.82**
17	0.80	**0.81**	**0.81**	**0.81**	**0.81**
21	0.77	0.78	0.78	**0.79**	0.78
Len.	consensusABC (Karaboga and Aslan 2016a)	ABC_{MOD1}	ABC_{MOD2}	$PABC_{MOD1}$	$PABC_{MOD2}$
10	0.90	**0.91**	**0.91**	**0.91**	**0.91**
11	0.89	**0.90**	**0.90**	**0.90**	**0.90**
14	**0.85**	**0.85**	0.83	**0.85**	**0.85**
21	0.78	0.78	0.78	**0.79**	0.78

Bold values show the best results of the given algorithms

González-Álvarez 2010, 2011; Karaboga and Aslan 2016a). When taking into account these situations, best similarity values for each motif of the serial ABC_{MOD1} by setting the *limit* parameter to ∞, parallel ABC_{MOD1} by setting the *limit* parameter to 500, serial and parallel ABC_{MOD2} algorithms by setting the *limit* parameter to 50 can be compared with the best similarity values obtained by the MOGAMOD, DEPT, MOABC and consensusABC algorithms.

Performance of the algorithms in terms of convergence characteristics and obtained results with them depend on the used data set and length of motifs. Because of these reasons, comparisons between algorithms are made according to the types of data sets. We first analyzed the performances of the algorithms for the *hm03r* data set. From the results given in Table 7, it is clearly seen that both serial and parallel implementations of the ABC_{MOD1} and ABC_{MOD2} algorithms outperform MOGAMOD, DEPT, MOABC and consensusABC algorithms or show similar performance on the discovery of motifs in the *hm03r* data set. While the highest difference on the similarity values of 9 nucleotide lengths motifs is equal to 0.09 between the MOGAMOD algorithm and ABC_{MOD2}, it is found to be 0.06 between MOABC and ABC_{MOD2}, 0.05 between DEPT and ABC_{MOD2} and 0.04 between consensusABC and ABC_{MOD2}. The effect of the improvements on the best similarity values obtained by the proposed algorithms can be seen more directly on the level of correctly matched dominant residue types in the produced alignment matrices. The alignment matrices produced by the serial and parallel ABC_{MOD2} algorithms on the *hm03r* data set for 9 nucleotide length contain at least 9 or more matches than the MOGAMOD, 6 or more matches than the MOABC, 5 or more matches than the DEPT, 4 or more matches than the consensusABC algorithm when the dominant residue types are concerned.

Secondly, the comparison results of the algorithms are given in the Tables 8 and 9 for the *yst04r* and *yst08r* data sets. When these tables are examined together, it can bee seen that serial and parallel ABC_{MOD1} and ABC_{MOD2} algorithms produce better results than the MOGAMOD, MOABC, DEPT and consensusABC algorithms as in the *hm03r* data set. In addition to this, there is not a remarkable difference between the results obtained by the ABC_{MOD1} and ABC_{MOD2} algorithms and both serial and parallel implementations of ABC_{MOD1} and ABC_{MOD2} are capable of finding motifs with the the near similarity values. While the highest difference on the similarity values of 11 nucleotide lengths motifs is equal to 0.13 between the MOGAMOD algorithm and ABC_{MOD1}, it is found to be 0.05 between MOABC and ABC_{MOD1} and 0.02 between DEPT and ABC_{MOD1} for the *yst08r* data set. For obtaining the mentioned improvements, the alignment matrices produced by the serial and parallel ABC_{MOD1} algorithms on the *yst08r* data set for 11 nucleotide length motifs contain at least 16 or more matches than the MOGAMOD, 6 or more matches than the MOABC and 3 or more matches than the DEPT algorithm when the dominant residue types are concerned.

5 Conclusion

In this work, we proposed two different ABC based motif discovery algorithms called ABC_{MOD1} and ABC_{MOD2} by employing the original definitions of the employed, onlooker and scout bee phases. The performances of the ABC_{MOD1} and ABC_{MOD2} algorithms have been compared with that of other population based metaheuristics including MOGAMOD, DEPT, MOABC and consensusABC algorithms over three data sets extracted from the TRANSFAC database to identify possible transcription-factor binding sites. The results obtained by the experimental studies showed that ABC_{MOD1} and ABC_{MOD2} algorithms are capable of finding more conserved regions that are likely to play a vital role in regulation of transcription or translation than the mentioned population based algorithms. We also investigated the parallelized implementations of the proposed algorithms. From the experimental studies, it is concluded that the parallelization does not only decrease running time of the algorithms, it also produces similar or better results than their sequential counterparts.

ABC_{MOD1} and ABC_{MOD2} algorithms have performed specific characteristics based on the selected data set and the length of motif being discovered. This type of difference is important to present new motif discovery techniques by combining new solution generation mechanisms of the ABC_{MOD1} and ABC_{MOD2} algorithms. In the future, more deterministic neighbour selection schemas can be used to increase convergence speed of the proposed algorithms and multiobjective adaptation of the ABC_{MOD1}, ABC_{MOD2} and their improved variations can be studied. It is also noted that ABC_{MOD1} and ABC_{MOD2} algorithms can be further investigated by applying different parallelization model, migration topologies and intervals.

References

Akay B, Karaboga D (2010) Artificial bee colony algorithm for large-scale problems and engineering design optimization. J Intell Manuf 23(4):1001. https://doi.org/10.1007/s10845-010-0393-4

Akay B, Karaboga D (2015) A survey on the applications of artificial bee colony in signal, image, and video processing. SIViP 9(4):967. https://doi.org/10.1007/s11760-015-0758-4

Alshamlan HM et al (2015) Genetic Bee Colony (GBC) algorithm: a new gene selection method for microarray cancer classification. Comput Biol Chem 56:49. https://doi.org/10.1016/j.compbiolchem.2015.03.001

Badem H, Basturk A, Caliskan A, Yuksel ME (2017) A new efficient training strategy for deep neural networks by hybridization of artificial bee colony and limited memory BFGS optimization algorithms. Neurocomputing 266(Supplement C). https://doi.org/10.1016/j.neucom.2017.05.061. http://www.sciencedirect.com/science/article/pii/S0925231217309487

Bolaji AL et al (2013) Artificial bee colony algorithm, its variants and applications: a survey. J Theor Appl Inf Technol 47(2)

Bulyk ML et al (2004) Computational prediction of transcription-factor binding site locations. Genome Biol 5(1):201

Cao B et al (2015) MOEPGA: a novel method to detect protein complexes in yeast protein–protein interaction networks based on multiobjective evolutionary programming genetic algorithm. Comput Biol Chem 58:173

Celik M et al (2016) CoABCMiner: an algorithm for cooperative rule classification system based on Artificial Bee Colony. Int J Artif Intell Tools 25(01):1. https://doi.org/10.1142/S0218213015500281

Chan TM, Leung KS, Lee KH (2012) Memetic algorithms for de novo motif discovery. IEEE Trans Evolut Comput 16(5):730

Chang BCH et al (2004) Particle swarm optimisation for protein motif discovery. Genet Program Evolvable Mach 5(2):203. https://doi.org/10.1023/B:GENP.0000023688.42515.92

Che D et al (2005) MDGA: motif discovery using a genetic algorithm. In: Proceedings of the 7th annual conference on genetic and evolutionary computation (ACM), pp 447–452

Das MK, Dai HK (2007) A survey of DNA motif finding algorithms. BMC Bioinform 8(Suppl 7):S21

González-Álvarez DL et al (2010) Solving the motif discovery problem by using differential evolution with pareto tournaments. In: IEEE congress on evolutionary computation (CEC), pp 1–8

González-Álvarez DL et al (2011) Finding motifs in DNA sequences applying a multiobjective Artificial Bee Colony (MOABC) algorithm. In: evolutionary computation, machine learning and data mining in bioinformatics. Springer, Berlin, pp 89–100

González-Álvarez DL et al (2012) Comparing multiobjective Artificial Bee Colony adaptations for discovering DNA motifs. In: Evolutionary computation, machine learning and data mining in bioinformatics. Springer, Berlin, pp 110–121

Huo H et al (2010) Optimizing genetic algorithm for motif discovery. Math Comput Modelling 52(11):2011

Jones NC, Pevzner P (2004) An introduction to bioinformatics algorithms. MIT press, London

Karaboga D, Aslan S (2015) A new emigrant creation strategy for parallel Artificial Bee Colony algorithm. In: IEEEE 2015 9th international conference on electrical and electronics engineering (ELECO), pp. 689–694

Karaboga D, Aslan S (2016a) A discrete Artificial Bee Colony algorithm for detecting transcription factor binding sites in DNA sequences. Genet Mol Res 15(02):1. https://doi.org/10.4238/gmr.15028645

Karaboga D, Aslan S (2016b) Best supported emigrant creation for parallel implementation of Artificial Bee Colony algorithm. IU J Electr Electron Eng 16(2):2055

Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: Artificial Bee Colony (ABC) algorithm. J Global Optim 39(3):459. https://doi.org/10.1007/s10898-007-9149-x

Karaboga D, Basturk B (2008) On the performance of Artificial Bee Colony (ABC) algorithm. Appl Soft Comput 8(1):687. https://doi.org/10.1016/j.asoc.2007.05.007

Kaya M (2009) MOGAMOD: multi-objective genetic algorithm for motif discovery. Expert Syst Appl 36(2, Part 1): 1039. https://doi.org/10.1016/j.eswa.2007.11.008

Lawrence CE, Reilly AA (1990) An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences, Proteins Struct Funct Bioinform 7(1):41. https://doi.org/10.1002/prot.340070105

Lesk A (2013) Introduction to bioinformatics. Oxford University Press, Oxford

Li N, Tompa M (2006) Analysis of computational approaches for motif discovery. Algorithms Mol Biol 1(1):1. https://doi.org/10.1186/1748-7188-1-8

Li B et al (2015) A balance-evolution artificial bee colony algorithm for protein structure optimization based on a three-dimensional AB off-lattice model. Comput Biol Chem. https://doi.org/10.1016/j.compbiolchem.2014.11.004

Liu FFM et al (2004) FMGA: finding motifs by genetic algorithm. In: Proceedings of the fourth IEEE symposium on bioinformatics and bioengineering (BIBE) 2004, pp 459–466. https://doi.org/10.1109/BIBE.2004.1317378

Liu J et al (2013) Heuristic-based tabu search algorithm for folding two-dimensional AB off-lattice model proteins. Comput Biol Chem 47:142. https://doi.org/10.1016/j.compbiolchem.2013.08.011

Luo Jw, Wang T (2010) Motif discovery using an immune genetic algorithm. J Theor Biol 264(2):319

Martinez E et al (2010) Compact cancer biomarkers discovery using a swarm intelligence feature selection algorithm. Comput Biol Chem 34(4):244. https://doi.org/10.1016/j.compbiolchem.2010.08.003

Mathe C et al (2002) Current methods of gene prediction, their strengths and weaknesses. Nucleic Acids Res 30(19):4103. https://doi.org/10.1093/nar/gkf543

Matys V et al (2003) TRANSFAC: transcriptional regulation, from patterns to profiles. Nucleic Acids Res 31(1):374. https://doi.org/10.1093/nar/gkg108

Ozturk C, Aslan S (2016) A new artificial bee colony algorithm to solve the multiple sequence alignment problem. Int J Data Min Bioinform 14(4):332

Shao L, Chen Y (2009) Bacterial foraging optimization algorithm integrating tabu search for motif discovery. In: IEEE international conference on bioinformatics and biomedicine BIBM'09 , pp 415–418

Tompa M et al (2005) Assessing computational tools for the discovery of transcription factor binding sites. Nat Biotechnol 23(1):137. https://doi.org/10.1038/nbt1053

Wang Z et al (2004) A brief review of computational gene prediction methods. Genom Proteom Bioinform 2(4):216

Zvelebil M, Baum J (2007) Understanding bioinformatics. Garland Science, New York