Mathematical Biosciences and Engineering, 16 (3): 1575–1596.
DOI: 10.3934/mbe.2019075
Received: 22 November 2018
Accepted: 27 January 2019
Published: 26 February 2019

http://www.aimspress.com/journal/MBE

Research article

A multi-objective imperialist competitive algorithm (MOICA) for finding motifs in DNA sequences

Saeed Alirezanejad Gohardani[1], Mehri Bagherian[1,*] and Hamidreza Vaziri[2]

[1] Department of Applied Mathematics, Faculty of Mathematical Science, University of Guilan, Rasht, Iran
[2] Department of Biology, Faculty of Science, University of Guilan, Rasht, Iran

* **Correspondence:** Email: mbagherian@guilan.ac.ir; Tel: +98-912-126-8190; Fax: +98-133-333-3509.

Abstract: Motif discovery problem (MDP) is one of the well-known problems in biology which tries to find the transcription factor binding site (TFBS) in DNA sequences. In one aspect, there is not enough biological knowledge on motif sites and on the other side, the problem is NP-hard. Thus, there is not an efficient procedure capable of finding motifs in every dataset. Some algorithms use exhaustive search, which is very time-consuming for large-scale datasets. On the other side, metaheuristic procedures seem to be a good selection for finding a motif quickly that at least has some acceptable biological properties. Most of the previous methods model the problem as a single objective optimization problem; however, considering multi-objectives for modeling the problem leads to improvements in the quality of obtained motifs. Some multi-objective optimization models for MDP have tried to maximize three objectives simultaneously: Motif length, support, and similarity. In this study, the multi-objective Imperialist Competition Algorithm (ICA) is adopted for this problem as an approximation algorithm. ICA is able to simulate more exploration along the solution space, so avoids trapping into local optima. So, it promises to obtain good solutions in a reasonable time. Experimental results show that our method produces good solutions compared to well-known algorithms in the literature, according to computational and biological indicators.

Keywords: motif discovery; Imperialist Competitive Algorithm; multi-objective optimization; DNA sequences; transcription factor binding site

1. Introduction

The information about motifs provides significant knowledge about evolutionary processes and the complexities of different organisms simultaneously. Biologists believe that some special proteins called Transcription Factors (TFs) bind to some patterns of the DNA substrings called Transcription Factor Binding Sites (TFBSs). Then, the process of gene expression initiates [1]. During this process, genes are transcribed into RNA and get activated or deactivated. These regulatory sites in DNA strings correspond to some conservative sequence patterns which are called *motifs*. The regulatory sites or TFBSs are called *occurrences*. Finding these regulatory sites or TFBSs seems to be difficult; however, discovering them helps molecular biologists to investigate the interaction between DNA and proteins, gene regulation, cell development, and cell reaction under physiological and pathological conditions. Occurrences have a fixed length, but they have slightly different compositions from their own motif.

This problem is one of the well-known problems in molecular biology and since it has been proven to be NP-hard [2], various methods and algorithms have been proposed to solve it. There are two classes of procedures for finding motifs: Pattern driven and sample driven approaches. In pattern driven procedures, the methods search on all $|\Sigma|^l$ candidate motifs. In sample-driven approaches patterns related to the given instance are only explored [3]. Methods to solve MDP can also be classified into two main groups: exact and approximate methods. The exact methods are very time consuming while approximate methods give a good result(s) in a reasonable time. Thus, they have attracted more attention from research communities. Heuristic procedures (as approximate procedures) have been widely used in the literature since they are fast and give a reasonable answer(s) [4]. Numerous existing algorithms identify motifs with a given length, but they are either not applicable or not efficient when searching motifs with different lengths. Finding motifs with and without gapped in the given sequences is the goal of many algorithms. Locating gapped motifs is a time-consuming task due to using combinatorial approaches [5]. Our proposed procedure is for a fixed-length motif without gap, and we don't study extensible-length motifs here.

To find real biological motifs, several assumptions which are close to the nature of the motif have been considered. Kaya [6] has listed some of these assumptions:

1. Each DNA sequence has an occurrence.
2. Each DNA sequence does not have an occurrence. This is more close to the definition of the motif in nature.
3. Sequences in the dataset have more than one occurrence.
4. There is more than one motif for sequences.

For each of these assumptions, a suitable algorithm has been proposed, but we are looking for an algorithm that behaves more similar to the nature of the motif, indeed. In fact, a composition of these assumptions can help us to find a reasonable answer(s). In the literature, the proposed procedures for solving MDP have been classified into two main groups: Statistical techniques and string-based methods [7]. Some significant statistical algorithms are the expectation maximization (EM) algorithm [8] and Gibbs Sampling [9] proposed by Lawrence and Raly which use a greedy approach. These algorithms use elaborated statistical techniques for finding motifs. MEME (Multiple EM for Motif Elicitation) is an extended EM algorithm for discovering motifs in unaligned biopolymer sequences [10]. AlignAce (Aligns Nucleic Acid Conserved Elements) finds occurrences

in a set of DNA sequences [11]. This algorithm tries to find profile motif. An important assumption taken into account by this procedure is that the motif model is one that is most different from the background motif model. Thus, they try to maximize the likelihood ratio of the motif model to the background model or the information entropy of the motif model in order to find the answer. In fact, log likelihood and information content are two scoring functions which are used for profile motif models. In MotifHyades, a novel probabilistic model was proposed and with two derived optimization algorithms attempted to find paired motifs in DNA sequences in human cell lines [12]. In [13], the k-spectrum modeling was used to catch DNA motif patterns from protein sequences. In the algorithm, multiple evaluation metrics gathered on millions of k-mer binding intensities from 92 proteins across 5 DNA-binding families.

The second class of algorithms is string-based ones. This kind of algorithms directly tries to find a motif through a substring that starts from zero to l. In fact, a tree of depth l is constructed where a node at depth k represents a k-length prefix of the motif. Some of the algorithms in this category are SPELLER [3], Weeder [14], MITRA [15], CENSUS [16], GENMOTIF [17], and RISOTTO [18]. Swarm intelligence algorithms are created based on natural phenomena. Genetic, Firefly, artificial bee colony and gravitation search algorithms are among the metaheuristic algorithms which are inspired by nature. Swarm intelligence deals with a system consisting of a group of individuals who move independently, but the group stays together and gradually converges to the optimal solution(s). These algorithms are iterative methods in which a single answer tries to improve itself during a period. Most of the metaheuristic algorithms in MDP use a single objective function, for example, MEME, MITRA, Weeder, and Consensus [19]. Kaya proposed MOGAMOD [6] considering three objective functions and demonstrated that considering the problem of MDP as a multi-objective optimization problem gives better motifs than the single objective ones. In fact, MOGAMOD attempts to optimize the length, support, and similarity of the motif simultaneously. These functions are in conflict with each other. Thus finding a solution that optimizes all these objectives at the same time is practically impossible and bounds to use the Pareto optimality notion. It is shown that the Multi-objective Artificial Bee Colony (MOABC) algorithm obtained the best results for a group of datasets among different approaches in the literature [20]. Similar to MOGAMOD, MOABC uses three objectives to find motifs. The authors of this paper also proposed a parallel version of this algorithm for protein strings [21]. Like all multi-objective optimization problems, it finds multiple answers at the end. In fact, this characteristic makes multi-objective algorithms superior to single objective ones. In this paper, we propose a multi-objective imperialist competition algorithm (MOICA) for MDP considering the three above mentioned objectives. Imperialist Competition Algorithm has recently attracted the attention of researchers to solve the optimization problems more efficiently.

For comparison purposes, we have used several measurements to demonstrate the efficiency of MOICA. We used hypervolume [22] which is an indicator that defines the volume of search space dominated by Pareto fronts. We also computed biological indicators such as nCC, nPC, nSn, nPPV, sSn, sPPV, sASP, and nSp to demonstrate the biological quality of our results [23]. Finally, we considered the motifs extracted from the same instances as in [6,7,20].

This paper is structured as follows: As you read the first section is the introduction. In Section2, we briefly define the motif discovery problem (MDP). Then in Section 3 describes the multi-objective version of MDP. Next, in Section 4, we describe the ICA and our algorithm that would adjust ICA for the multi-objective version of MDP. Next section shows the experimental

results obtained from MOICA and compares the results with some of the already existing approximation algorithms. Finally, we present our conclusion in the last section.

2. Materials and methods

Consider n sequences with length m, which are a composition of the alphabet Σ, are called by s_i. We defined $s_i[k, l]$ as a subsequence of s_i with length l that started from k. Also, let $s_i[k]$ denote a single alphabet of s_i in position k. In this paper, we will be using the terms string(s) and sequence(s) interchangeably. A motif is obtained in the given input sequences by its occurrences. We denoted the occurrences related to one motif as a vector $x = (x_1, x_2, \dots, x_n)$, where x_i is the starting position of an occurrence in the s_i and $x_i = 0,1,2, \dots, m - l + 1$, where l is the length of each occurrence in the dataset. Let us suppose that each sequence has a probability of selection. Then, we indicate $x_i = 0$ if we don't select any occurrence in s_i. As a result, there is not an occurrence in the i^{th} sequence with $x_i = 0$ (Figure 1). The probability of selection is called *rate* in our paper. It is clear that a sequence with a rate = 40 has less chance for selection than one with a rate = 60. The start of metaheuristic algorithms is to create individual populations. Using each of these occurrences as an individual, then, we extract occurrences in the sequences regarding the *length of motif* (Figure 1). The length of the motif in our algorithm varies between 7 and 64 as in [24]. In order to obtain a *consensus motif*, we put all of the occurrences under each other (Figure 1). A composition of the maximum alphabet frequency for each column results in consensus motif (Figure 1). We define two variables to show consensus in a precise way: $w_i = \max f_i^{\Sigma}$ is the maximum alphabet frequency, and the corresponding alphabet for w_i is denoted by c_{w_i}, both of them extracted from the i^{th} column of the occurrence matrix. Then, the consensus motif is expressed through $consensus(x) = c_{w_1} c_{w_2} \dots c_{w_n}$.

After finding the consensus motif, we find *concordance* of each occurrence; it is a normalized value that shows the resemblance between the consensus motif and the occurrence (Figure 1).

$$concordance(x_i) = \frac{l - d_H\big(s_i[x_i, l], consensus(x)\big)}{l}, \quad i = 1,2, \dots, n, x_i \neq 0$$

Where d_H is the Hamming distance between two sequences with equal length, i.e., the number of positions where the two sequences differ. Furthermore, same as other algorithms in the literature, the threshold of our algorithm for concordance is 0.5. It means that if the concordance of i^{th} occurrence is less than 0.5, we let $x_i = 0$. The number of $x_i > 0$ is the *support* of the motif which depends on its occurrences. The mathematical notation of it can be written in the following way:

$$support(x) = \sum_{i=1}^{n} u_{x_i}$$

Where $u_{x_i} = \begin{cases} 1 & x_i > 0 \\ 0 & x_i = 0 \end{cases}$. In our example of Figure 1, a single motif occurrence exists in each sequence at most, but in reality, it can be more than one. On the other side, we make a limitation on the support, $support(x) \geq \delta,\ \delta = \begin{cases} 2 & n \leq 4 \\ 3 & n > 4 \end{cases}$. Next, concept *similarity* is employed, which involve the maximum value from the i^{th} column of the position weight matrix.

$$similarity(x) = \frac{\sum_{i=1}^{l} \frac{w_i}{support(x)}}{l}$$

Then, the similarity is the average of maximum frequency for a potential motif with length l (Figure 1).

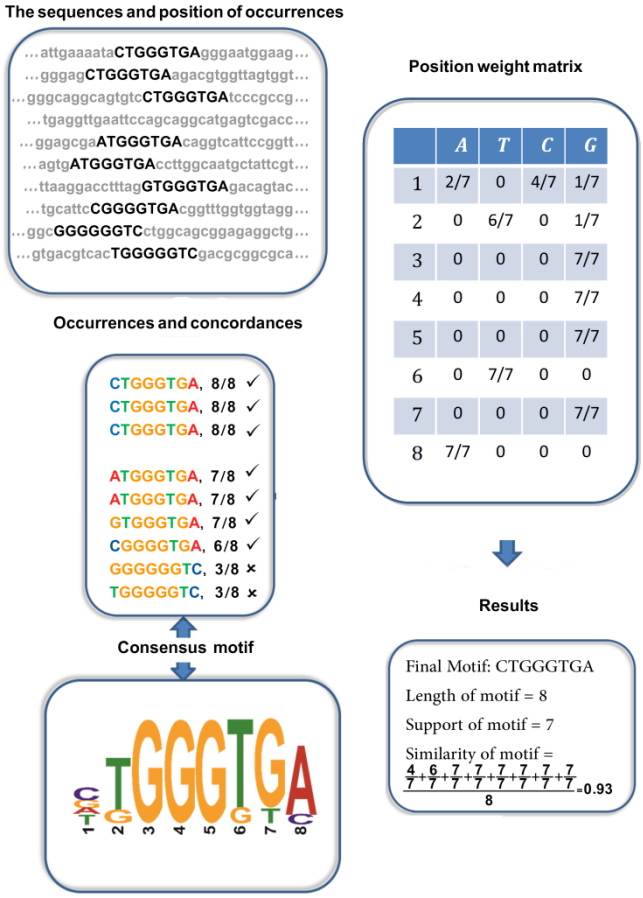

Figure 1. An example which represents an individual $x = [308, 6, 314, 0, 468, 371, 184, 238, 391, 447]$ in a given ten sequences. As shown, nine of ten sequences have occurrence. Consensus motif is the maximum alphabet frequency in the i[th] column of occurrences. Seven occurrences from nine occurrences obtained the threshold of 50% in the concordance; i.e., $support(x) = 7$. We computed the position weight matrix and similarity. Then we reported the results: Length, support, and similarity.

We computed the similarity of the example motif in Figure 1. A concept that we used in our algorithm is *Complexity* based on Fogel [25]. The complexity of a string with length l in $|\Sigma|$ different alphabets is computed as follows:

$$Complexity(x_i) = \log_{|\Sigma|} \frac{l!}{n_{i,1}!\, n_{i,2} \ldots n_{i,|\Sigma|}!}, \quad i = 1,2, \ldots, n$$

Where $n_{i,j}$ is the number of the j^{th} alphabet from Σ in the selected sequence x_i and $|\Sigma| = 4$ in DNA. As an example, the complexity of string "AAAAAAA' is $\log_4 \frac{7!}{7!0!0!0!} = 0$, and the complexity of string "AAATAAA" is $\log_4 \frac{7!}{6!1!0!0!} = 1.40$. These short strings are somehow useless in biology since $n_A = 7{,}6$. The complexity of an individual x is the average complexity value of each sequence.

$$complexity(x) = \frac{\sum_{i=1}^{n} complexity(x_i)}{support(x)}$$

The complexity of two strings "AAAAAAA" and "AAATAAA" is $\frac{0+1.40}{2} = 0.70$. The complexity depends on the length of the motif, which we normalize it based on the elicited motif length. In our algorithm, we took a minimum threshold of 0.5 (50%) for the complexity.

Our goal in this paper is to maximize the length, support, and similarity of the motif simultaneously while avoiding falls in low complexity result(s). In fact, these three values (length, support, and similarity) are our objectives. Fitness value of an individual is assessed based on its length, support, and similarity. We also have used the fitness value of each individual to compute the fitness value of a group of individuals. These two values help us to find the weakest and the most likehood individuals and the groups of individuals in our method.

We gathered all the variables and functions to define the problem in Table 1, again. The definition of each one also is illustrated.

Table 1. Notations and definitions.

Variable	Definition
$\boldsymbol{n}$	Number of sequences
$\boldsymbol{m}$	Length of each sequence
$\boldsymbol{l}$	length of motif
$\boldsymbol{s_i}$	i^{th} sequence with length m which is a composition of the alphabet Σ
$\boldsymbol{s_i[k,l]}$	A subsequence from s_i with length l that started from k
$\boldsymbol{s_i[k]}$	A single alphabet of s_i in position k
$\boldsymbol{x_i}$	The starting position of an occurrence in the s_i and $x_i = 0{,}1{,}2, \dots, m-l+1$
$\boldsymbol{x}$	An individual, $x = (x_1, x_2, \dots, x_n)$
$\boldsymbol{f_i^{\Sigma}}$	Frequency of Σ in the i^{th} column of the occurrence matrix
$\boldsymbol{w_i}$	$w_i = \max f_i^{\Sigma}$
$\boldsymbol{c_{w_i}}$	The corresponding alphabet for w_i
$\boldsymbol{n_{i,j}}$	Number of the j^{th} alphabet from Σ in the selected sequence x_i
$\boldsymbol{u_{x_i}}$	$u_{x_i} = \begin{cases} 1 & x_i > 0 \\ 0 & x_i = 0 \end{cases}$
$\boldsymbol{\delta}$	$\delta = \begin{cases} 2 & n \le 4 \\ 3 & n > 4 \end{cases}$
$\boldsymbol{consensus(x)}$	$consensus(x) = c_{w_1} c_{w_2} \dots c_{w_n}$

Continued on next page

Variable	Definition
$\boldsymbol{concordance(x_i)}$	$concordance(x_i) = \frac{l - d_H\left(s_i[x_i, l], consensus(x)\right)}{l}, \ i = 1{,}2, \dots, n, x_i \neq 0$

$\boldsymbol{support(x)}$	$support(x) = \sum_{i=1}^{n} u_{x_i}$
$\boldsymbol{similarity(x)}$	$similarity(x) = \frac{\sum_{i=1}^{l} \frac{\mathrm{w_i}}{support(x)}}{l}$
$\boldsymbol{Complexity(x_i)}$	$Complexity(x_i) = \log_{\lvert\Sigma\rvert} \frac{l!}{n_{i,1}!\, n_{i,2} \dots n_{i,\lvert\Sigma\rvert}!},\ i = 1,2,\dots,n$
$\boldsymbol{complexity(x)}$	$complexity(x) = \frac{\sum_{i=1}^{n} complexity(x_i)}{support(x)}$

We can express the problem mathematically as:

$\mathrm{Max}\, f(x) = \big(l, support(x), similarity(x)\big)$

s.t.

$support(x) \geq \delta$

$complexity(x) \geq 0.5$

$concordance(x_i) \geq 0.5,\ \ i = 1,2,..,n, x_i \neq 0$

$x_i = 0, 1, 2, \dots, m - l + 1,\ \ i = 1,2, \dots, n$

$l = 7, 8, \dots, 64$

3. Multi-objective problems

A single-objective optimization problem is defined as:

$$Max\ f(x)$$

s.t

$$g_j(x) \leq 0,\ j \in J$$

$$h_k(x) = 0,\ k \in K$$

Where $f(x)$ is a single function and $g_j(x), h_k(x)$ are its constraints. In multi-objective optimization problems (MOOP), we have a vector of functions which has to be optimized. In such problems, the aim is to find the best value for all components, though the components of multi-objective function may be in conflict with each other, such that increasing in one component may causes decreasing in some other component(s).

The general form of a MOOP can be written as:

$$Max\ f(x) = \big(f_i(x)\big),\ i = 1, \dots, n$$

s.t

$$g_j(x) \leq 0,\ j \in J$$

$$h_k(x) = 0,\ k \in K$$

Where $f(x)$ is a vector with n components and $g_j(x), h_k(x)$ are its constraints. After solving a MOOP, we obtain multiple solutions which may not be better than other solutions. We call such solutions as *non-dominated* solutions. In fact, the value of x *dominate*s the value of y, which we

indicate as $f(x) \succ f(y)$ if and only if $f(x)$ is better than $f(y)$ in at least one component, which may also be not worse than the other components. We say x is *pareto- optimal* or *non-dominated optimal* if there is no y in feasible space such that $f(y)$ dominates $f(x)$. We call a set of pareto-optimal solutions or non-dominated optimal solutions as *pareto-optimal set*. In fact, after solving a MOOP, we have a pareto-optimal set [26]. We also call the value of the pareto-optimal set as *pareto fronts*. Our goal is to find the best pareto fronts.

A lot of metaheuristic algorithms based on natural phenomena have been defined to solve optimization problems like genetic algorithm inspired by gene evolution, ant colony algorithm mimiced by ant search behavior, fish swarm optimization based on fish behavior looking for food. Among the metaheuristic algorithms, ICA has attracted the researchers' attention due to its ability to find good solutions for NP-hard problems. In the next section, we explain the ICA in detail. This is the main part of our paper.

In swarm algorithms, a group of individuals work collaboratively to solve a problem. To start the algorithm, initial individuals (countries) are created randomly. The cost of each individual (country) is calculated on the basis of its objective function value. We can refer to ICA ability to simulate a wider exploration during the solution searching space as an advantage.

4. The MOICA algorithm

The Imperialist Competition Algorithm proposed by Atashpaz et al. in 2007 [27] was inspired by the intelligent behaviour of imperialist competition in the real world. This algorithm has some simple parameters such as the number of countries, the number of imperialists, and the maximum number of iterations. To solve multi-objective MDP, we propose a new algorithm based on the Imperialist Competition Algorithm. Similar to the other evolutionary algorithms, the first part of this algorithm is the creation of initial population. Then, all countries are classified into two groups: Imperialists and colonies. In fact, some of the best countries are selected as imperialists and the rest of them are called colonies. Colonies are divided by imperialists on the basis of their power. Thus, a group of empires are created which challenge each other to catch more countries. Two main processes occur after this *assimilation* process: *Intra competition* and *extra completion*. In intra competition, each colony moves towards its own imperialist and the imperialist tries also to increase its power. During this process, a colony replaces its own imperialist if it has more power. The next process is extra competition. In this process, the algorithm finds the weakest colony in the weakest empire and adds it to one of the strongest empires. The total power of an empire depends on the power of its imperialist and a percentage of the mean power of its colonies.

These two main processes would repeat in a reasonable time until the empires gradually collapse and finally all countries converge into one empire.

In this work, a particular intelligent behaviour which causes countries to rise or collapse is considered. Individuals in this algorithm are similar to MOGAMOD [6] and MOABC [20]. The main steps of the algorithm are given below:

Algorithm MOICA:

1. *Input*: n sequences, the selection probability (rate), the number of Population (nPop), the number of Empires (nEmp), and the Maximum Iteration (MaxIt).
2. Create nPop countries with a fixed length $l \in [l_{min}, l_{max}]$ and compute support and similarity for each of them.

3. Use non-dominated sorting to sort all Motifs (countries).
4. *Assimilation*: Select nEmp of best Motifs (countries) as imperialists and divide the other nCol countries between them based on imperialists' positions in Step 3.
5. *Intra Competition*: Make a revolution in colonies and imperialists in each empire and replace each imperialist by a colony if it is better.
6. Use non-dominated sorting to sort all Motifs (countries).
7. Compute the fitness function of each empire and find the weakest.
8. *Extra competition*: Select the weakest Motif (country) from the weakest empire and give it to the empire that has the most likehood to possess it.
9. Repeat 5–8 MaxIt times.
10. *Output*: Non-dominated set.

MaxIt, n sequences, rate, nPop, and nEmp are given as input. We should know the minimum and maximum length of motifs, l_{min} and l_{max} respectively to search within this range. To start the algorithm, we create and evaluate the initial countries (item 2). This step repeats for each motif length. Here, we add two propositions to our algorithm in this line. The first proposition is that the support of each individual (country) cannot be less than 2 for $n = 2,3$ and it cannot be less than 3 for $n \geq 4$. The second proposition is that we don't let an individual (country) with low complexity get produced. In fact, we require the program to reproduce a new individual instead of a previous one which didn't satisfy our two propositions. The main process of MOICA starts after item 2.

After non-dominate sorting (in item 3), all the countries assimilate into nEmp countries which are selected after the sorting process (in item 4). We can divide the rest of algorithm into three blocks: Intra competition, non-dominate sorting, and extra competition. In intra competition block, we create colonies on the basis of their distances from their own imperialists and also create a new imperialist (we explain the way of mutation and crossover at the end of this section). We replace the imperialist by each of its colonies or create a new imperialist if its support and similarity is better than the imperialist based on the dominant concept and if its complexity is not low (item 5). In the second block, we do non-dominated sorting to find the rank between all countries (item 6). Hence, we calculate the fitness value of each country by:

$$(nPop - 1)rank, \;\; rank = 1,2, \dots, nPop.$$

We use the fitness value of each country to calculate the fitness (total power) of an empire. The fitness value of an empire is the fitness value of its imperialist added to a ζ percent of average fitness value of its colonies. In our proposal, we take $\zeta = 30\%$.

In the last block, we do an extra competition to find the weakest colony in the weakest empire to add it to one of the strongest empires (items 7 and 8). We find both of them based on their fitness, the fitness value of empire, and the fitness value of each country. Finally, we repeat this process Maximum Iteration (MaxIt) times to reach to some non-dominated imperialists. Then, we report them as motifs.

At the end of this section, we explain how we make new colonies move towards an imperialist. We also explain how we mutate a country (imperialist or colony). Let $Imp = (a_1, a_2, \dots, a_n)$ and $Col = (b_1, b_2, \dots, b_n)$. First of all, in order to make a Col move towards an Imp for the creation of a new colony $(b_1', b_2', \dots, b_n')$ we do as follows:

$$b_i^{'} = \begin{cases} 0 & a_i = 0, \eta < p_{rev} \\ b_i & a_i = 0, \eta > p_{rev} \\ a_i & a_i \neq 0, b_i = 0' \\ b_i + \lceil \beta\eta(a_i - b_i) \rceil & a_i \neq 0, b_i \neq 0 \end{cases}$$

Where $\eta \sim U(0,1)$ that is a uniform number between 0 and 1. We take $\beta = 2$ and $p_{rev} = 10\%$ in MOICA. We also notice that if $a_i \neq 0$, $b_i \neq 0$ then we don't let $b'_i < 1$ or $b'_i > m - l + 1$. If $b_i < 1$ or $b_i > m - l + 1$, we replace it with one and $m - l + 1$, respectively.

Mutation is to make a revolution in the country (imperialist or colony). We produce a new country $(a'_1, a'_2, \ldots, a'_n)$ as follows:

$$a_i^{'} = \begin{cases} 0 & a_i = 0 \\ a_i + \lceil \sigma\theta \rceil & a_i \neq 0 \end{cases}$$

Here we take $\theta \sim U(-1,1)$ and $\sigma = 0.1(m - l + 1)$. We also notice that if $a_i \neq 0$, then we don't let $a'_i < 1$ or $a'_i > m - l + 1$. If $a'_i < 1$ or $a'_i > m - l + 1$, we replace it with one and $m - l + 1$, respectively.

5. Experimental results

In this section, we compare our algorithm with other significant algorithms in the literature. In order to show the efficiency of our algorithm, we performed some experiments as in Kaya [6] and Gonzalez et al. [7,20], and on real dataset as TRANSFAC database [23]. We compare our algorithm with some previous significant methods in the literature such as, MEME [10], Weeder [14], AlignACE [11], MOABC [20], MO-VNS [28], DEPT [24], MO-FA [29], MO-GSA [30], SPEA2 [31], and NSGA-II [32]. The comparison is made according to three metrics: Hypervolume indicator, biological indicators, and motif finding. Our proposed algorithm is written in C++[1]. The computation has been performed on a laptop computer with an Intel(R) Core(TM) i3 CPU M 330 (2.13 GHz) and 4.0 GB memory.

One important performance measure for comparing MOICA with other multi-objective algorithms for finding TFBs is hypervolume indicator [22]. Hypervolume was used by Gonzalez [20] to show that MOABC mostly produces better results than the other algorithms in the literature. Hypervolume is an indicator that indicates the volume of search space covered by the result of algorithm. To calculate the hypervolume, (0,0,0) is selected as the worst point. In order to obtain more accurate results, we ran our algorithm 31 times for each instance to show its statistical significance and reported its average and standard deviation in Table 3. Reference volume in each instance depends on the number of sequences, the maximum length of each sequence, and the maximum similarity that is 1 (100%). For example, a dataset with 4 sequences will have the reference volume of 4 × 64 × 1. Since other source codes are not available, to compare MOICA with algorithms of [6,7], we have approximately used the same runtimes. To assure the power of our algorithm, we selected a composition of 54 real, generic, and Markov chain from TRANSFAC dataset [23] (called as ending in "r", "g", and "m" respectively) with different sequence numbers and lengths.

[1] The source code would be available upon request.

The dataset specifications, algorithm parameters for each instance, and runtime(s) are reported in Table 2. The total size of each data is equal to the product of the number of sequence and the length of sequence. As it is seen, MOICA has 4 parameters: the probability of selection of each sequence (rate), the Maximum Iteration (MaxIt), the number of Population (nPop), and the number of Empires (nEmp). To reach the best result, we ran the program several times to find the best parameters for MOICA and reported all of them in Table 2.

Table 2. MOICA parameters and runtime for each instance.

Dataset	Seq.	Len.	rate	MaxIt	nPop	nEmp	nCol	Time(sec.)
yst05r	3	500	55%	390	160	50	110	15
yst02g	4	500	55%	390	160	50	110	15
yst10m	5	1000	55%	390	160	50	110	20
yst07m	6	500	55%	390	160	50	110	20
yst06g	7	500	55%	390	160	50	110	20
yst03m	8	500	55%	390	160	50	110	25
yst01g	9	1000	55%	390	160	50	110	25
yst08r	11	1000	55%	390	160	50	110	30
yst09g	16	1000	55%	330	135	35	100	30
yst04r	7	1000	70%	390	160	50	110	25
yst03r	8	500	70%	390	160	50	110	25
yst09g	16	1000	70%	300	125	35	90	30
mus11r	12	500	60%	400	160	50	110	30
mus07r	4	1500	75%	400	160	50	110	15
hm16r	7	3000	60%	470	180	50	130	30
hm04r	13	2000	60%	400	160	50	110	40
dm07m	3	1500	55%	390	160	50	110	15
dm08m	3	2000	80%	420	170	50	120	15
dm03m	3	2000	80%	420	170	50	120	15
dm05g	3	2500	55%	390	160	50	110	15
dm01g	4	1500	80%	420	170	50	120	20
dm04g	4	2000	80%	420	170	50	120	20
hm12r	2	500	90%	420	170	50	120	15
hm25g	2	500	90%	420	170	50	120	15
hm14r	2	1000	90%	420	170	50	120	15
hm05r	3	1000	80%	420	170	50	120	15
hm23r	4	500	80%	420	170	50	120	20
hm15r	4	2000	55%	390	160	50	110	15
hm19g	5	500	55%	390	160	50	110	20
hm21g	5	500	55%	390	160	50	110	20
hm07m	5	1000	55%	390	160	50	110	20
hm18m	5	3000	55%	390	160	50	110	20
hm22m	6	500	55%	390	160	50	110	20

Continued on next page

Dataset	Seq.	Len.	rate	MaxIt	nPop	nEmp	nCol	Time(sec.)
hm10m	6	500	55%	390	160	50	110	20

hm13r	6	1000	55%	390	160	50	110	20
hm16g	7	3000	55%	350	155	45	110	20
hm24m	8	500	55%	345	150	40	110	20
hm11g	8	1000	55%	345	150	40	110	20
hm06g	9	500	55%	345	150	40	110	20
hm26m	9	1000	55%	345	150	40	110	20
hm02r	9	1000	55%	345	150	40	110	20
hm03r	10	1500	55%	390	160	50	110	30
hm09g	10	1500	55%	390	160	50	110	30
hm17g	11	500	55%	390	160	50	110	30
hm04m	13	2000	55%	340	140	40	100	30
hm08m	15	500	55%	330	135	35	100	30
hm01g	18	2000	55%	300	130	30	100	30
hm20r	35	2000	55%	190	110	20	90	30
mus09r	2	500	55%	390	160	50	110	10
mus01r	3	500	55%	390	160	50	110	15
mus12m	3	500	55%	390	160	50	110	15
mus06g	3	500	55%	390	160	50	110	15
mus08m	3	1500	55%	390	160	50	110	15
mus05r	4	500	55%	390	160	50	110	15
mus07g	4	1500	55%	390	160	50	110	15
mus03g	5	500	55%	390	160	50	110	20
mus04m	7	1000	55%	350	155	45	110	20
mus02r	9	1000	55%	345	150	40	110	20
mus11m	12	500	55%	345	150	40	110	30
mus10g	13	1000	55%	340	140	40	100	30
dm05r	5	2500	95%	450	160	50	110	20
dm04r	4	2000	80%	450	200	50	150	25
dm01r	4	1500	75%	400	160	50	110	15

Hypervolume indicator is the first indicator used to compare MOICA with MOABC [20], MO-VNS [28], DEPT [24], MO-FA [29], MO-GSA [30], SPEA2 [31], and NSGA-II [32].

The result of this comparison is shown in Table 3. We divide the results in four parts: Datasets with $n = 2$ sequences, datasets with $n = 3, 4, 5, 6, 7$ sequences, datasets with $n = 8, 9$ sequences, and datasets with $n > 9$ sequences. In datasets with $n = 2$ sequences, NSGA-II has the best hypervolume value among all algorithms which is specified in Table 2, though MOABC, MO-GSA and SPEA2 have a value close to that of NSGA-II. In datasets with $n = 2, 3, 4, 5, 6, 7$ sequences, MOABC gets the best results in different cases most of the time. MO-VNS, NSGA-II, and SPEA2 have close results in this interval. For example, NSGA-II has the best hypervolume in dm05g ($n = 2$), mus07g ($n = 3$), dm01g ($n = 3$), and yst04r ($n = 7$). On the other hand, MO-VNS's hypervolume and SPEA2's hypervolume decrease quickly for $n > 5$ and $n > 4$, respectively. DEPT, MO-FA, MO-GSA, and MOICA have very weak hypervolume in this interval in most cases, though MO-FA and MOICA have the best hypervolume for hm16g ($n = 7$) and yst10m ($n = 5$), respectively. DEPT, MO-FA and MOICA gets better hypervolume if we increase the number of sequences.

For datasets with $n = 8, 9$ sequences, DEPT has the best hypervolume in most cases. Also, MOICA and MO-FA have a value close to that of DEPT in this interval, but other algorithms have very weak results. For datasets with $n > 9$, MOICA has the best results and its hypervolume for hm10g ($n = 18$) and hm20r ($n = 35$) is significant. Other algorithms are very weak for datasets with more than 9 sequences except MO-FA that has a better result just for hm09g ($n = 10$).

Table 3. The mean and standard deviation of each Algorithm hypervolume on 54 datasets.

Dataset	n, m	MOICA	MOABC	MO-VNS	DEPT	NSGA-II	MO-FA	MO-GSA	SPEA2
mus09r	2,500	83.12%	92.95%	84.79%	90.95%	93.46%	90.36%	92.92%	93.35%
		0.002	0.005	0.061	0.002	0.001	0.004	0.005	0.002
hm12r	2,500	85.12%	94.29%	90.69%	91.50%	94.71%	91.14%	93.88%	94.59%
		0.002	0.005	0.034	0.005	0.002	0.006	0.005	0.007
hm25g	2,500	81.93%	93.66%	79.92%	91.30%	93.94%	90.24%	93.20%	93.82%
		0.003	0.007	0.062	0.008	0.010	0.007	0.011	0.003
hm14r	2,1000	84.07%	93.88%	86.77%	91.40%	94.40%	91.03%	94.11%	93.96%
		0.002	0.006	0.062	0.003	0.004	0.006	0.003	0.003
yst05r	3,500	82.32%	86.89%	86.61%	80.76%	85.96%	83.63%	81.29%	85.66%
		0.004	0.007	0.007	0.021	0.010	0.014	0.020	0.004
mus01r	3,500	80.28%	86.93%	85.90%	80.40%	85.26%	84.44%	80.50%	85.65%
		0.004	0.006	0.007	0.017	0.011	0.017	0.022	0.004
mus12m	3,500	80.46%	85.78%	85.41%	80.04%	84.19%	82.72%	80.78%	84.96%
		0.004	0.006	0.006	0.018	0.012	0.010	0.018	0.004
mus06g	3,500	79.60%	84.33%	84.21%	80.01%	83.35%	81.91%	79.94%	83.46%
		0.003	0.006	0.007	0.020	0.010	0.008	0.017	0.003
hm05r	3,1000	82.38%	85.50%	85.41%	80.46%	85.19%	82.73%	80.32%	85.10%
		0.005	0.009	0.007	0.019	0.012	0.009	0.015	0.005
dm07m	3,1500	81.65%	86.09%	85.87%	81.01%	85.01%	83.53%	80.42%	85.19%
		0.004	0.011	0.007	0.015	0.008	0.010	0.020	0.006
mus08m	3,1500	80.71%	85.77%	85.11%	80.85%	84.52%	83.10%	80.48%	84.60%
		0.005	0.008	0.007	0.010	0.010	0.010	0.023	0.004
dm08m	3,2000	81.81%	85.96%	85.75%	81.30%	85.90%	83.25%	80.94%	85.17%
		0.003	0.008	0.007	0.015	0.007	0.010	0.017	0.004
dm03m	3,2000	81.29%	86.06%	85.56%	81.38%	84.56%	83.52%	80.75%	85.16%
		0.003	0.011	0.007	0.017	0.007	0.011	0.021	0.006
dm05g	3,2500	82.18%	86.39%	85.88%	81.55%	86.69%	83.67%	80.30%	85.59%
		0.003	0.010	0.008	0.014	0.005	0.012	0.021	0.005
yst02g	4,500	78.80%	82.87%	82.50%	78.21%	79.81%	80.82%	78.78%	82.28%
		0.003	0.010	0.007	0.016	0.007	0.011	0.012	0.005
hm23r	4,500	77.16%	82.38%	81.57%	78.13%	79.51%	80.89%	78.23%	81.18%
		0.002	0.007	0.007	0.018	0.007	0.011	0.013	0.005
mus05r	4,500	77.88%	81.69%	81.45%	77.56%	79.81%	80.25%	78.08%	80.44%
		0.003	0.008	0.006	0.019	0.006	0.008	0.013	0.004

Continued on next page

Dataset	n, m	MOICA	MOABC	MO-VNS	DEPT	NSGA-II	MO-FA	MO-GSA	SPEA2
mus07g	4,1500	85.55%	89.21%	89.08%	80.01%	90.43%	82.43%	79.85%	87.73%

		0.013	0.029	0.015	0.034	0.022	0.015	0.018	0.010
dm01g	4,1500	80.91%	83.24%	82.95%	79.66%	83.77%	81.62%	79.03%	81.74%
		0.004	0.008	0.006	0.017	0.010	0.013	0.015	0.004
dm04g	4,2000	80.34%	83.77%	82.90%	79.78%	81.16%	81.91%	79.31%	81.08%
		0.003	0.014	0.008	0.018	0.006	0.014	0.014	0.005
hm15r	4,2000	82.80%	88.59%	85.23%	80.29%	85.23%	83.98%	80.23%	85.93%
		0.010	0.027	0.016	0.022	0.022	0.042	0.025	0.009
hm19g	5,500	76.76%	81.37%	80.35%	78.71%	79.33%	80.08%	76.81%	79.74%
		0.003	0.009	0.007	0.022	0.007	0.009	0.019	0.004
mus03g	5,500	72.74%	79.84%	77.39%	77.61%	76.48%	79.67%	74.58%	77.31%
		0.003	0.008	0.018	0.016	0.012	0.012	0.044	0.004
yst10m	5,1000	75.53%	66.67%	66.08%	63.49%	65.24%	64.99%	63.39%	65.38%
		0.003	0.007	0.005	0.014	0.004	0.009	0.010	0.003
hm21g	5,1000	75.83%	81.12%	79.76%	78.72%	78.24%	79.49%	77.01%	78.61%
		0.002	0.010	0.007	0.012	0.008	0.010	0.019	0.004
hm07m	5,1000	74.87%	81.24%	79.66%	78.91%	79.39%	79.91%	77.37%	78.71%
		0.003	0.009	0.006	0.017	0.005	0.010	0.017	0.004
hm18m	5,3000	75.16%	81.31%	79.38%	79.80%	77.58%	79.97%	77.39%	78.58%
		0.003	0.008	0.007	0.014	0.004	0.009	0.023	0.004
yst07m	6,500	72.27%	78.27%	74.22%	76.81%	74.57%	77.36%	70.49%	75.92%
		0.003	0.008	0.032	0.012	0.013	0.008	0.038	0.005
hm22m	6,500	72.22%	77.91%	74.12%	76.98%	75.02%	76.84%	69.87%	75.53%
		0.003	0.007	0.028	0.011	0.030	0.009	0.047	0.005
hm10m	6,500	72.06%	78.23%	72.36%	77.11%	74.61%	77.54%	69.01%	75.22%
		0.003	0.011	0.034	0.014	0.020	0.009	0.038	0.005
hm13r	6,1000	72.93%	78.18%	72.88%	77.24%	76.08%	77.35%	70.27%	75.19%
		0.003	0.010	0.031	0.008	0.010	0.008	0.040	0.005
yst06g	7,500	74.10%	75.93%	72.04%	74.94%	73.84%	75.75%	68.05%	73.19%
		0.002	0.012	0.028	0.018	0.011	0.014	0.030	0.005
yst04r	7,1000	75.30%	75.54%	72.88%	73.50%	77.04%	73.98%	68.20%	71.43%
		0.003	0.012	0.028	0.020	0.007	0.014	0.034	0.006
mus04m	7,1000	70.02%	71.10%	65.49%	74.31%	67.86%	74.24%	63.99%	69.43%
		0.003	0.023	0.021	0.012	0.012	0.010	0.030	0.020
hm16g	7,3000	76.90%	81.55%	70.36%	78.42%	71.82%	83.63%	69.12%	71.13%
		0.015	0.055	0.026	0.044	0.015	0.051	0.043	0.023
yst03m	8,500	69.97%	69.68%	65.03%	73.26%	66.44%	72.65%	62.59%	68.37%
		0.004	0.024	0.026	0.012	0.011	0.010	0.038	0.011
hm24m	8,500	69.08%	67.65%	62.65%	72.91%	65.10%	72.19%	59.58%	68.28%
		0.003	0.019	0.028	0.012	0.011	0.008	0.036	0.013
hm11g	8,1000	72.17%	69.93%	63.86%	74.54%	69.38%	73.15%	60.14%	69.34%
		0.005	0.021	0.028	0.021	0.031	0.017	0.044	0.012
hm06g	9,500	69.52%	63.21%	59.43%	70.68%	60.06%	69.37%	56.60%	61.24%
		0.003	0.019	0.019	0.027	0.015	0.012	0.029	0.012

Continued on next page

Dataset	n, m	MOICA	MOABC	MO-VNS	DEPT	NSGA-II	MO-FA	MO-GSA	SPEA2
yst01g	9,1000	69.94%	65.16%	61.48%	70.72%	62.75%	70.14%	59.37%	63.16%

		0.004	0.025	0.021	0.014	0.013	0.011	0.024	0.013
hm26m	9,1000	68.51%	64.43%	60.98%	70.68%	60.86%	70.48%	58.89%	62.28%
		0.004	0.025	0.025	0.016	0.013	0.012	0.031	0.015
hm02r	9,1000	68.68%	63.45%	59.04%	70.85%	61.50%	69.88%	57.57%	59.09%
		0.006	0.033	0.022	0.023	0.014	0.014	0.024	0.013
mus02r	9,1000	69.08%	63.58%	59.34%	70.02%	61.99%	69.78%	57.21%	60.62%
		0.005	0.024	0.025	0.019	0.013	0.015	0.025	0.012
hm03r	10,1500	69.23%	60.18%	58.75%	65.03%	51.59%	69.06%	58.44%	53.77%
		0.005	0.026	0.014	0.030	0.038	0.020	0.021	0.017
hm09g	10,1500	68.28%	59.23%	57.72%	65.20%	50.71%	68.71%	57.51%	54.79%
		0.005	0.027	0.017	0.033	0.043	0.023	0.022	0.013
hm17g	11,500	66.08%	55.55%	55.12%	64.12%	49.94%	65.86%	54.40%	52.47%
		0.006	0.023	0.019	0.033	0.037	0.030	0.024	0.012
yst08r	11,1000	71.15%	62.91%	63.03%	67.80%	64.92%	69.12%	61.24%	57.96%
		0.006	0.023	0.030	0.048	0.028	0.019	0.030	0.014
mus11m	12,500	62.98%	51.25%	49.54%	58.85%	45.32%	62.01%	49.51%	48.87%
		0.007	0.020	0.031	0.039	0.043	0.014	0.031	0.014
hm04m	13,2000	63.88%	53.20%	52.21%	60.90%	45.03%	62.17%	52.18%	47.89%
		0.005	0.024	0.022	0.030	0.029	0.018	0.029	0.011
mus10g	13,1000	62.64%	48.92%	46.22%	55.34%	41.73%	59.25%	47.05%	43.94%
		0.005	0.021	0.030	0.039	0.036	0.016	0.032	0.010
hm08m	15,500	61.25%	48.57%	46.81%	57.15%	42.63%	58.07%	46.98%	43.45%
		0.009	0.023	0.037	0.046	0.028	0.015	0.037	0.011
yst09g	16,1000	62.11%	47.92%	45.74%	55.70%	42.68%	56.86%	46.50%	42.34%
		0.007	0.022	0.037	0.056	0.022	0.021	0.029	0.011
hm01g	18,2000	59.96%	42.05%	34.40%	42.98%	33.92%	49.56%	37.95%	34.55%
		0.010	0.033	0.029	0.054	0.027	0.065	0.042	0.009
hm20r	35,2000	46.70%	25.08%	22.67%	25.33%	22.89%	36.43%	25.68%	21.91%
		0.016	0.020	0.027	0.026	0.014	0.065	0.020	0.006

In biological indicators, we examine the biological properties of our proposal with fifteen methods in TRANSFAC database. In this assessment, we just ran MOICA 5 times for each dataset to elicit its properties. This database contains a list of four species with different n and m. We first define these biological tools based on both the nucleotide level and the site level. Specifically, at the nucleotide level [33].

- nTP is the number of nucleotide positions in both known sites and predicted sites;
- nFN is the number of nucleotide positions in known sites but not in predicted sites;
- nFP is the number of nucleotide positions not in known sites but in predicted sites;
- nTN is the number of nucleotide positions in neither known sites nor predicted sites.

At the site level [33]:

- sTP is the number of known sites overlapped by predicted sites;
- sFN is the number of known sites not overlapped by predicted sites;
- sFP is the number of predicted sites not overlapped by known sites.

We ran MOICA on the selected TRANSFAC database and found the following results for Fly, Human, Mouse, and Yeast.

Table 4. The result of MOICA in the motif-finding assessment.

MOICA	nTP	nFP	nFN	nTN	sTP	sFP	sFN
Fly	177	655	494	41674	11	7	40
Human	1092	4860	4027	277021	53	49	245
Mouse	612	1692	1027	52169	36	14	62
Yeast	339	2349	871	57441	21	23	53

Like Table 4, these parameters were computed for AlignAce, ANN-Spec, Consensus, GLAM, Improbizer, MEME, MEME3, MITRA, MotifSampler, ologo/dyad-analysis, QuickScore, SeSiMCMC, Weeder, and YMF. They were reported in Tompa [33].

We need some biological indicators to show the performance of each method on different species. These biological indicators are defined at the nucleotide ($x = n$) and site ($x = s$) levels as follows [33]:

- $Sensitivity: xSn = \frac{xTP}{xTP+xFN}$
- $Positive\ Predictive\ Value: xPPV = \frac{xTP}{xTP+xFP}$
- $Specificity: nSp = \frac{nTN}{nTN+nFP}$
- $Performance\ Coefficient: nPC = \frac{nTP}{nTP+nFN+nFP}$
- $the\ Correlation\ Coefficient: nCC = \frac{nTP.nTN-nFN.nFP}{\sqrt{(nTP+nFN)(nTN+nFP)(nTP+nFP)(nTN+nFN)}}$
- $average\ performance: sASP = \frac{sSn+sPPV}{2}$

Add nTP, nFP, nFN, nTN, sTP, sFP, and sFN over the TRANSFAC database and compute each indicator based on the top values (Table 3). We can see that our algorithm improves all biological indicators for Mouse and Fly instances significantly. It also gets best results for Human instances except for nPPV indicator. We also ran our proposal on Yeast datasets, but its result is weak. Table 4 indicates that Weeder has the best performance on Yeast. These biological indicators show that MOICA gets good results in most instances. It is important to mention that some biological tools achieve good results just in some instances but fail to obtain reasonable motifs in others.

Table 5. The comparison of biological indicators (nCC, nSn, sSn, nPC, nPPV, sPPV and sASP) of Human, Mouse, Yeast and Fly by different algorithms.

Fly	nSn	nPPV	nPC	nCC	sSn	sPPV	sASP
MOICA	0.263785	0.21274	0.133484	0.223421	0.215686	0.611111	0.413399

AlignAce	0	0	0	−0.0063762	0	0	0
ANN-Spec	0.0253353	0.0175258	0.010468	0.0023548	0.0196078	0.009434	0.0145209
Consensus	0	0	0	−0.0110554	0	0	0
GLAM	0.0029806	0.004902	0.001857	−0.0084518	0	0	0
Improbizer	00149031	0.0176056	0.0081367	0.0018679	0.0196078	0.0227273	0.0211676
MEME	0.0417288	0.0421053	0.0214067	0.0267982	0.0588235	0.0555556	0.0571895
MEME3	0.0372578	0.0263713	0.0156838	0.0130431	0.0588235	0.0447761	0.0517998
MITRA	0	0	0	−0.0078616	0	0	0
MotifSampler	0.0044709	0.0082192	0.0029042	−0.0055135	0	0	0
ologo/dyad-analysis	0	0	0	−0.0149773	0	0	0
QuickScore	0	0	0	−0.0157195	0	0	0
SeSiMCMC	0.1013413	0.0538827	0.0364611	0.0537034	0.0980392	0.125	0.1115196
Weeder	0.0119225	0.0344828	0.0089385	0.0112184	0.0196078	0.0344828	0.0270453
YMF	0	0	0	−0.0138211	0	0	0

Human	nSn	nPPV	nPC	nCC	sSn	sPPV	sASP
MOICA	0.213323	0.183468	0.10943	0.182113	0.177852	0.519608	0.34873
AlignAce	0.0392655	0.102551	0.0292236	0.0531478	0.0738255	0.1235955	0.0987105
ANN-Spec	0.090252	0.1031711	0.0505747	0.0812784	0.1644295	0.0983936	0.1314116
Consensus	0	NaN	0	NaN	0	NaN	NaN
GLAM	0.0236374	0.0367669	0.0145977	0.0155035	0.0402685	0.06	0.0501342
Improbizer	0.0416097	0.0476084	0.0227079	0.0284205	0.0704698	0.0483871	0.0594284
MEME	0.0380934	0.0603902	0.0239176	0.0343922	0.0604027	0.0810811	0.0707419
MEME3	0.0420004	0.0470975	0.0227057	0.0282219	0.0637584	0.0788382	0.0712983
MITRA	0.0244188	0.0471342	0.0163484	0.0214655	0.0402685	0.046875	0.0435717
MotifSampler	0.0250049	0.0416531	0.015873	0.0188157	0.0469799	0.0430769	0.0450284
ologo/dyad-analysis	0.0371166	0.213964	0.0326629	0.0825991	0.0604027	0.15	0.1052013
QuickScore	0.0050791	0.0099388	0.0033727	-0.0056328	0	0	0
SeSiMCMC	0.0459074	0.0279962	0.0176984	0.0134837	0.0671141	0.0630915	0.0651028
Weeder	0.0543075	0.2747036	0.047497	0.1154935	0.1073826	0.2580645	0.1827235
YMF	0.0410236	0.0967296	0.029661	0.0521165	0.0738255	0.080292	0.0770587

Mouse	nSn	nPPV	nPC	nCC	sSn	sPPV	sASP
MOICA	0.373398	0.265625	0.183729	0.290236	0.367347	0.720000	0.543673
AlignAce	0.028676	0.0490605	0.0184314	0.0152885	0.0306122	0.0361446	0.0333784
ANN-Spec	0.0433191	0.0430564	0.0220703	0.0139802	0.0816327	0.0425532	0.0620929
Consensus	0.0488103	0.1062417	0.0346021	0.0531418	0.1020408	0.1219512	0.111996
GLAM	0.0073215	0.0187793	0.0052957	−0.0068546	0.0102041	0.0153846	0.0127943
Improbizer	0.1079927	0.1219008	0.0607412	0.0894309	0.2244898	0.1605839	0.1925369
MEME	0.0732154	0.1337793	0.0496689	0.0789261	0.1428571	0.175	0.1589286

Continued on next page

Mouse	nSn	nPPV	nPC	nCC	sSn	sPPV	sASP
MEME3	0.1061623	0.170088	0.0699357	0.1137754	0.1938776	0.2	0.1969388
MITRA	0.0061013	0.015456	0.0043937	−0.0090299	0.0204082	0.0327869	0.0265975
MotifSampler	0.0445394	0.0913642	0.0308668	0.0441428	0.0816327	0.0952381	0.0884354
ologo/dyad-	0.0268456	0.1067961	0.0219233	0.03947	0.0612245	0.0952381	0.0782313

analysis							
QuickScore	0.0359976	0.089939	0.0263864	0.0390251	0.0816327	0.0677966	0.0747146
SeSiMCMC	0.0622331	0.0408818	0.0252976	0.0145461	0.1020408	0.0952381	0.0986395
Weeder	0.0616229	0.1753472	0.0477767	0.0882065	0.122449	0.1791045	0.1507767
YMF	0.1012813	0.2017011	0.0722997	0.1247729	0.2040816	0.1818182	0.1929499
Yeast	nSn	nPPV	nPC	nCC	sSn	sPPV	sASP
MOICA	0.280165	0.126116	0.0952515	0.163648	0.283784	0.477273	0.380528
AlignAce	0.1855754	0.1849758	0.1020954	0.1684257	0.28	0.2019231	0.2409615
ANN-Spec	0.165316	0.14011099	0.0820595	0.1331542	0.3066667	0.1411043	0.2238855
Consensus	0.0794165	0.2	0.0602706	0.1149074	0.1466667	0.2391304	0.1928986
GLAM	0.0713128	0.0585106	0.0332075	0.0432325	0.1466667	0.0578947	0.1022807
Improbizer	0.1572123	0.0950049	0.0629461	0.0988463	0.2666667	0.1333333	0.2
MEME	0.1928687	0.3801917	0.1467324	0.260354	0.3066667	0.3833333	0.345
MEME3	0.2098865	0.3001159	0.140914	0.2381559	0.32	0.3037975	0.3118987
MITRA	0.1110211	0.1525612	0.0686717	0.1148955	0.16	0.1538462	0.1569231
MotifSampler	0.2560778	0.5039872	0.2045307	0.3501753	0.3866667	0.4915254	0.439096
ologo/dyad-analysis	0.0899514	0.3303571	0.0760795	0.1639418	0.1866667	0.3043478	0.2455072
QuickScore	0.0534846	0.0613953	0.0294249	0.0391636	0.12	0.0434783	0.0817391
SeSiMCMC	0.1012966	0.0570255	0.0378673	0.0504601	0.0933333	0.0721649	0.0827491
Weeder	0.2925446	0.5340237	0.2330536	0.3863337	0.52	0.5492958	0.5346479
YMF	0.1442464	0.3296296	0.1115288	0.2076962	0.28	0.3387097	0.3093548

We also compared our proposal with MOABC in biological realm. The nTP, nFP, nFN, nTN, sTP, sFP, and sFN is not available for MOABC except a figure in [7] where nTP, nFP, nFN, nTN, sTP, sFP, and sFN of all species added up and then all biological indicators computed. We did not the same rule because it compares the mean results and may be unfair.

The last experiment is accomplished based on motif finding and its non-dominated solutions are reported in Table 6. We ran our algorithm on four datasets of dm04g, hm22m, mus03g, and yst01g; hm22m and mus03g have 6 and 5 sequences respectively and the sequence length of each one is 500 bps, dm04g has 4 sequences, each with length 2000 bps, and yst01g has 9 sequences each with length 1000 bps. We compared MOICA with MOABC [20], because MOABC almost had the best performance among all algorithms.

We used the reported results for these four datasets in [7]. We computed length, support, similarity, and complexity of the reported results for MOABC and compared them with our results. The comparative results are recorded in Table 6. Our goal and other researchers were not to maximize complexity, but we report it just to show a fair comparison. In fact, most of algorithms can find the best results for length, support, and similarity with a low complexity. Table 6 shows that MOICA has the best results. We almost fixed support, length, and complexity to find a motif with high similarity. The results confirm that MOICA beats MOABC in motif finding.

Notice that MOABC and MOICA are multi-objective, so we can find alternative answers with different support and similarity for each length. As we can see in Table 6, MOICA obtained the best results for all of these datasets.

In this experiment, we ran our algorithm so that only one or two empire(s) remained at the end. To reach this goal, we fixed the parameters of our algorithm at nPop = 200, nEmp = 50, nCol = 150

and MaxIt = 800. We also defined rate based on datasets. The selection probability of each sequence (rate) depends on the data and the kind of the motif that we want to find. For example, if the support of the desired motif is close to the number of sequences of the dataset, it is better to take rate from 70 to 100. Moreover, the selection probability of each sequence (rate) changes from 30 to 100.

Table 6. The comparison of the conserved motifs predicted by MOABC and MOICA methods for dm04g, hm22m, mus03g and yst01g data.

data	Method	support	length	similarity	complexity	Predicted Motif
dm04g	MOABC	3	33	0.64	0.84	TCAACTGTAAATATAACTTAAAAAGGGAATACT
dm04g	MOICA	3	33	0.74	0.85	GTTTGGAAGTGCTTAAATAAACTTGCAAAAAAC
hm22m	MOABC	4	33	0.61	0.81	ATGACCCACACCACGCGCACGCATGGCCCGGCC
hm22m	MOICA	4	34	0.68	0.83	CCACGCCAGACGGGCGTTGCAAGCCAGACCTACT
mus03g	MOABC	4	35	0.58	0.87	AAGGCGTTGCTCAAGTGTTAAGAAAATACTGACAC
mus03g	MOICA	4	36	0.63	0.87	TGGATAGAAGAACTACAACCTTCATGTCATACATTT
yst01g	MOABC	4	16	0.59	0.74	ATGAAATTAAACCCAA
yst01g	MOICA	4	17	0.69	0.74	TACCATAATTCATAGAT

6. Discussion and conclusion

In this paper, we have proposed a metaheuristic algorithm for solving MDP considering a Multi-objective Imperialist Competition Algorithm (MOICA) based on three optimization functions. A big contribution of our method is that it gives alternative answers in a single run. Thus, the user can select useful ones among multiple answers.

In order to assess the performance of our algorithm, we ran MOICA on 54 sequence datasets from four species with different length and various numbers of sequences and compared the result with those obtained by previous methods. To facilitate comparing, we used the same criteria as used in multi-objective algorithms like hypervolume indicator. It is demonstrated that our algorithm gets a significant average hypervolume indicator in a set of instances with more than nine sequences. We also computed biological indicators (nCC, nPC, nSn, nPPV, sSn, sPPV, sASP, and nSp) for some selected instances in the benchmark that were mentioned in previous works. Its biological indicators have the best results in all instances except yeast. Furthermore, we also found motifs for 4 different datasets with different number of sequences and lengths. It can be concluded that the solutions found by our algorithm gives the best results among all algorithms in the literature. In future, we can modify our algorithm by composing other SI algorithms to make it more realistic with a better performance. Artificial Bee Colony (ABC) is a metaheuristic procedure which obtained good result for MDP considering a multi-objective framework.

As future works, we intend to design new approaches to discover common patterns in sequences. At first, we interest in integrating ICA and ABC to invent a new algorithm to improve results for a bigger set of the datasets. Other desirable research is directly related to model this

problem as an optimization-programming problem to use the existing algorithms for solving it. Finally, we would like to try to find new methods with the aim of improving both optimization programming and biological results.

Financial support

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors. The authors appreciate University of Guilan for partial financial support.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. F. Zare-Mirakabad, H. Ahrabian and M. Sadeghi, et al., Genetic algorithm for dyad pattern finding in DNA sequences, *Genes Genet. Syst.,* **84** (2009), 81–93.
2. M. Li, B. Ma and L. Wang, Finding similar regions in many sequences, *J. Comput. Syst. Sci.*, **65** (2002), 73–96.
3. M. F. Sagot, Spelling approximate repeated or common motifs using a suffix tree, Springer, 1998.
4. F. W. Glover and G. A. Kochenberger, Handbook of metaheuristics, Springer Science & Business Media, 2006.
5. E. Czeizler, T. Hirvola and K. Karhu, A graph-theoretical approach for motif discovery in protein sequences, *IEEE/ACM Trans. Comput. Biol. Bioinf.*, **14** (2017), 121–130.
6. M. Kaya, MOGAMOD: Multi-objective genetic algorithm for motif discovery, *Expert. Syst. Appl.,* **36** (2009), 1039–1047.
7. D. L. González-Álvarez, M. A. Vega-Rodríguez and Á. Rubio-Largo, Multiobjective optimization algorithms for motif discovery in DNA sequences, *Genet. Program. Evolvable Mach.,* **16** (2015), 167–209.
8. C. E. Lawrence and A. A. Reilly, An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences, *Proteins,* **7** (1990), 41–51.
9. C. E. Lawrence, S. F. Altschul and M. S. Boguski, et al., Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment, *Science,* **262** (1993), 208–214.
10. T. L. Bailey and C. Elkan, Fitting a mixture model by expectation maximization to discover motifs in bipolymers, *Proc. Int. Conf. Intell. Syst. Mol. Biol.,* **2** (1994), 28–36..
11. F. P. Roth, J. D. Hughes and P. W. Estep, et al., Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole-genome mRNA quantitation, *Nat. Biotechnol.,* **16** (1998), 939–945.
12. K. C. Wong, MotifHyades: Expectation maximization for de novo DNA motif pair discovery on paired sequences, *Bioinformatics,* **33** (2017), 3028–3035.
13. K. C. Wong, DNA Motif Recognition Modeling from Protein Sequences, *iScience,* **7** (2018), 198–211.

14. G. Pavesi, P. Mereghetti and G. Mauri, et al., Weeder Web: Discovery of transcription factor binding sites in a set of sequences from co-regulated genes. *Nucleic Acids Res.*, **32** (2004), W199–W203.
15. E. Eskin and P. A. Pevzner, Finding composite regulatory patterns in DNA sequences, *Bioinformatics*, **18** (2002), S354–S363.
16. P. A. Evans and A. D. Smith, *Toward optimal motif enumeration*, Springer, 2003.
17. J. Serra, A. Matic and A. Karatzoglou, et al., *A genetic algorithm to discover flexible motifs with support*, IEEE, 2016.
18. N. Pisanti, A. M. Carvalho and L. Marsan, et al., *RISOTTO: Fast extraction of motifs with mismatches*, Springer, 2006.
19. G. Z. Hertz and G. D. Stormo, Identifying DNA and protein patterns with statistically significant alignments of multiple sequences, *Bioinformatics*, **15** (1999), 563–577.
20. D. L. González-Álvarez, M. A. Vega-Rodríguez and J. A. Gómez-Pulido, et al., *Finding Motifs in DNA Sequences Applying a Multiobjective Artificial Bee Colony (MOABC) Algorithm*, Springer, 2011.
21. D. L. González-Álvarez, M. A. Vega-Rodríguez and Á. Rubio-Largo, Searching for common patterns on protein sequences by means of a parallel hybrid honey-bee mating optimization algorithm, *Parallel. Comput.*, **76** (2018), 1–17.
22. E. Zitzler and L. Thiele, Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach, *IEEE T. Evolut. Comput.*, **3** (1999), 257–271.
23. E. Wingender, P, Dietze and H. Karas, et al., TRANSFAC: A database on transcription factors and their DNA binding sites, *Nucleic Acids Res.*, **24** (1996), 238–241.
24. D. L. González-Álvarez, M. A. Vega-Rodríguez and J. A. Gómez-Pulido, et al., *Solving the motif discovery problem by using differential evolution with pareto tournaments*, IEEE, 2010.
25. G. B. Fogel, D. G. Weekes and G. Varga, et al., Discovery of sequence motifs related to coexpression of genes using evolutionary computation, *Nucleic Acids Res.*, **32** (2004), 3826–3835.
26. E. Zitzler, K. Deb and L. Thiele, Comparison of multiobjective evolutionary algorithms: Empirical results, *Evolut. Comput.*, **8** (2000), 173–195.
27. E. Atashpaz-Gargari and C. Lucas, *Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition*, IEEE, 2007.
28. D. L. Gonzalez-Álvarez, M. A. Vega-Rodriguez and J. A. Gomez-Pulido, et al., Predicting DNA motifs by using evolutionary multiobjective optimization, *IEEE. T. Syst. Man. Cy. C.*, **42** (2012), 913–925.
29. X. S. Yang, *Firefly algorithms for multimodal optimization*, Springer, 2009.
30. D. L. González-Álvarez, M. A. Vega-Rodríguez and J, A. Gómez-Pulido, et al., *Applying a multiobjective gravitational search algorithm (MO-GSA) to discover motifs*, Springer, 2011.
31. E. Zitzler, M. Laumanns and L. Thiele, *SPEA2: Improving the strength Pareto evolutionary algorithm*, 2001.
32. K. Deb, A. Pratap and S. Agarwal, et al., A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE T. Evolut. Comput.*, **6** (2002), 182–197.
33. M. Tompa, N. Li and T. L. Bailey, et al., Assessing computational tools for the discovery of transcription factor binding sites, *Nat. Biotechnol.*, **23** (2005), 137.

AIMS Press