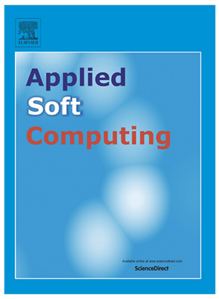
Multi-objective artificial bee colony for designing multiple genes encoding the same protein

Belen Gonzalez-Sanchez, Miguel A. Vega-Rodríguez,
Sergio Santander-Jiménez, José M. Granado-Criado



PII: S1568-4946(18)30580-5
DOI: https://doi.org/10.1016/j.asoc.2018.10.023
Reference: ASOC 5141

To appear in: *Applied Soft Computing Journal*

Received date : 29 April 2018
Revised date : 13 August 2018
Accepted date : 11 October 2018

Multi-Objective Artificial Bee Colony for Designing Multiple Genes Encoding the Same Protein

Belen Gonzalez-Sanchez[a], Miguel A. Vega-Rodríguez[b,*], Sergio Santander-Jiménez[c], José M. Granado-Criado[b]

[a]*Escuela Politécnica, Universidad de Extremadura, Avda. de la Universidad s/n, 10003 Cáceres, Spain.*
[b]*Instituto de Investigación en Tecnologías Informáticas Aplicadas de Extremadura (INTIA), Universidad de Extremadura, Avda. de la Universidad s/n, 10003 Cáceres, Spain.*
[c]*INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, 1000-029 Lisboa, Portugal.*

Abstract

The improvement of protein expression levels represents one of the most important goals in synthetic biology. In order to accomplish it, a promising and widely-used strategy lies on integrating multiple genes that encode the subject protein into an organism genome. This important task, however, is affected by several challenging issues. Firstly, the integration of highly similar sequences can potentially induce homologous recombination, a negative effect that implies a reduction in the number of genes effectively integrated. This is the reason why it is important to design multiple protein-coding sequences (also named CDSs) that are as different as possible, between both different CDSs and different subsequences within the same CDS. Additionally, codon usage frequencies in these CDSs should be as highly adapted to the organism as possible. Therefore, this task involves different and conflicting objectives that must be optimized, thus being suitable to be tackled as a multi-objective optimization problem. In this work, we design and implement the algorithm MOABC (Multi-Objective Artificial Bee Colony) to solve the problem of designing multiple CDSs that encode the same protein, considering three objectives to be optimized. The experimen-

*Corresponding author
Email addresses: `belengs@unex.es` (Belen Gonzalez-Sanchez), `mavega@unex.es` (Miguel A. Vega-Rodríguez), `sergio.jimenez@tecnico.ulisboa.pt` (Sergio Santander-Jiménez), `granado@unex.es` (José M. Granado-Criado)

tal evaluation herein performed suggests that MOABC is able to obtain relevant results, showing statistically significant improvements over the ones found in the literature.

Keywords: Multi-Objective Artificial Bee Colony, Design of Multiple Genes, Encoding of the Same Protein, Protein-Coding Sequence (CDS).

1. Introduction

The idea of maximizing the expression levels of proteins is becoming an increasingly relevant research goal in the field of synthetic biology. In this context, the integration of multiple genes encoding a certain protein into an organism genome represents a promising approach to achieve such a goal. Integrating n genes encoding the same protein implies, in general terms, that the expression levels of this protein can also increase n times (see for example [1]). Although this effect does not happen in absolutely all the cases [2], such strategies have been attracting increasing interest throughout the years [3, 4, 5].

However, this is a difficult task. The process of integrating multiple genes into an organism genome is complex and time-consuming, and also it involves high cost. In order to reduce all these problems, in the current protocols, the multiple genes are integrated very near each other within the organism genome [3, 4, 5]. This is a good solution, but it has a weak point. When repetitive sequences are very near each other, they can induce homologous recombination, and as a consequence, some of these sequences are lost [6]. For example, if a gene is replicated 4 times (g_1, g_2, g_3, g_4) and these copies are concatenated in tandem and integrated into a genome, a homologous recombination could take place between g_1 and g_4, and the copies of gene g_2 and g_3 would be lost.

For that reason, it is very important that the multiple protein-coding sequences (also named CDSs) are as different as possible, between both different CDSs and different subsequences within the same CDS. The exact length that the identical sequences must have in order to induce homologous recombination is not known, and it could change depending on the organism. For example, [7] reported, in *Escherichia coli*, that the identical sequences should have at least 23 bp (base pairs). [8] reported, in *Bacillus subtilis*, homologous recombination of identical sequences with a length of 70 bp. [9] reported, in *Saccharomyces cerevisiae*, a highly increased rate of homologous recombination with identical sequences of 30 bp. Although the exact length is not known, all the studies agree that the longer the identical sequences are, the higher the homologous recombination rate is. For this reason, our multi-objective optimization problem will have two objectives trying to minimize the length of the longest common subsequence found and trying to maximize the distance (difference) between the two most similar CDSs, respectively.

The way to obtain different CDSs that encode the same protein is using different codons for the corresponding amino acids. Almost all the amino acids can be encoded by different synonymous codons (a codon is a series of three nucleotides). Therefore, changing the synonymous codons used for encoding a particular amino acid we can obtain a different CDS that encodes the same protein. Synonymous codons occur with different frequencies in an organism, and the choice of codons may affect the expression levels of a protein [10]. For this reason, it is important to select the codons with the highest usage frequencies (the most highly adapted ones). This will be the third objective in our multi-objective optimization problem, in which we will try to maximize the minimum value of the Codon Adaptation Index (CAI). As we can observe, different and conflicting objectives have to be optimized.

Regarding the related work, codon usage frequency optimization in an individual CDS has been analyzed in recent studies [11, 12, 13], and also different tools have been proposed, such as COOL [14], D-Tailor [15] or OPTIMIZER [16]. In this regard, [17] demonstrated that the codon usage of highly expressed genes was selected in evolution to maintain the efficiency of global protein translation, and [18] shed insight into the factors that influenced the codon usage frequency in genes associated with the central nervous system. However, our approach is different from all these previous related works, because we also optimize sequences to increase the nucleotide differences among multiple CDSs that encode the same protein. In that sense, after a literature review, to our best knowledge, the only previous proposal that addresses the same multi-objective optimization problem has been recently published [19]. In that last paper, the problem was solved by using the NSGA-II algorithm. Therefore, we compare our results with this previous related work in order to show the advantages of our approach.

In particular, our approach is based on the Multi-Objective Artificial Bee Colony (MOABC) algorithm. This algorithm is a multi-objective adaptation of the Artificial Bee Colony (ABC) algorithm [20], which has been selected due to its good results in other applications [21]. In this work, we have adapted the ABC algorithm to the multi-objective context and we have designed and implemented the MOABC algorithm for designing a set of CDSs that encode a protein avoiding inducing homologous recombination and using the synonymous codons with the best CAIs. Apart from the MOABC algorithm designed and implemented here, currently, there are a number of MOABC algorithms available in the literature. In this sense, [22] developed a MOABC that used a grid-based approach to adaptively assess the Pareto front maintained in an external archive. The external archive was used to control the flying behaviors of the individuals and structuring the bee colony. [23] proposed a MOABC where an elite-guided solution generation strategy was designed to exploit the neighborhood of the existing solutions based on the guidance of the elite. Furthermore, a novel fitness calculation method was presented to calculate the selecting probability for onlooker bees. [24] designed a MOABC with dynamic population, which synergized the idea of extended life-cycle evolving model to balance the exploration and exploitation trade-off. In this approach, the bee was able to reproduce and die dynamically throughout the foraging process and population size varied as the algorithm ran. [25] developed a MOABC that integrated genetic operators. In this way, this approach used the traditional crossover and mutation offspring process.

To our best knowledge, this is the first time that a swarm intelligence algorithm is used to solve our multi-objective problem. As we will see, after a comparative study and the corresponding statistical evaluation, we can conclude that our approach obtains very good results. In fact, as a final contribution of our work, we can highlight that our approach obtains better results than the ones previously published in the literature.

The rest of this paper is organized as follows. Section 2 explains and gives a formal definition of the multi-objective optimization problem to solve. After that, section 3 details and describes our approach (MO-ABC) to address this optimization problem. Section 4 includes the experiments performed, the results obtained, and the comparisons with the results found in the literature. Finally, section 5 explains the conclusions of this work and indicates possible future lines.

2. Problem Definition

Given a protein to encode, a solution to our multi-objective optimization problem is a set of sequences (CDSs) encoding this protein. The number of CDSs per protein is determined by the user. Every CDS is a sequence of nucleotides, and therefore, we represent a solution to our problem as a set of sequences of characters. Given a protein, all its CDSs will have the same length. An example of solution is shown in Table 1.

The evaluation of each solution is based on three objective functions. The first function concerns the encoding of each codon (preferring the codons with the highest usage frequencies) and the other two functions are related to the avoidance of repetitions between CDSs and between subsequences within the same CDS. The following subsections explain in detail these three objective functions.

2.1. Codon Adaptation Index (CAI)

The first objective function is related with the occurrence frequency of each codon, because some codons have higher frequency than others, and therefore, it is better to select these better adapted codons. The focus in this function is the minimum value of Codon Adaptation Index ($mCAI$). Equation 1 shows how this calculation is made.

$$mCAI = \min_{1 \leq i \leq I} \quad CAI(CDS_i) \tag{1}$$

where CDS_i is each CDS that encodes the protein, I is the number of CDSs, and CAI value is calculated for each one as indicated by Equation 2.

$$CAI(CDS_i) = \sqrt[N]{\prod_{n=1}^{N} W(codon_{i,n})}, \tag{2}$$

where N is the number of codons that CDS_i has and W is the weight assigned to the $codon_{i,n}$. This weight is calculated as the usage frequency of $codon_{i,n}$ relative to (divided into) the usage frequency of the most frequent codon among the synonymous codons of $codon_{i,n}$ [26]. The usage frequencies have been obtained from the research carried out by [19].

The objective is to optimize the minimum CAI value among all the CDSs ($mCAI$). The use of the average CAI value would not be sufficiently appropriate because it is possible that there could be an i-th CDS with a very low CAI within a good average CAI. In conclusion, the objective is to maximize the minimum CAI value ($mCAI$) to achieve that all the CDSs have high codon adaptation index.

2.2. Hamming Distance between CDSs (HD)

The second objective function is focused on selecting CDSs that are very different between them. This function is based on a normalized Hamming distance value between two CDSs from a solution. The objective is calculated as the minimum value among all these pairs of CDSs (see Equation 3).

$$mHD = \min_{1 \leq i < j \leq I} \quad \frac{HD(CDS_i, CDS_j)}{L}. \tag{3}$$

For a pair of CDSs, CDS_i and CDS_j, both with length L nucleotides, the Hamming Distance (HD) is calculated as shown in Equation 4.

$$HD(CDS_i, CDS_j) = \sum_{1 \leq k \leq L} \sigma(CDS_{i,k}, CDS_{j,k}), \tag{4}$$

where the i-th and j-th CDSs are compared and, in both CDSs, the k-th nucleotide is evaluated. If $CDS_{i,k}$ and $CDS_{j,k}$ are equal then σ will be 0. However, if they are different nucleotides, σ is set to 1.

The objective is to maximize mHD. Again, we use the minimum value, as in the first objective function, because if we use the average value, we could have a very low HD within a good average.

2.3. Length of Repeated or Common Substrings (LRCS)

The third objective function is focused on the idea of selecting different subsequences between different CDSs and also within the same CDS. In conclusion, this objective tries to reduce the length of repeated or common substrings (LRCS).

We say that we find a common substring $S_{i,p,l}$ in the i-th CDS, at the p-th position and with a length of l characters (nucleotides), when the same or another CDS (j-th CDS) has the same substring $S_{j,q,l}$ at the same (in this case, $i \neq j$) or different q-th position. For example, in Table 1, *GUGUUA* is the longest common substring between all pairs of CDSs, although there are other repeated substrings, e.g. in CDS_3, *UGCU*, but they have a lower length. The objective is to minimize the maximum length of the repeated or common substring ($MLRCS$) found among CDSs (see Equation 5).

CDS_1	AGA	GUA	UU**G**	**UGU**	**UA**C
CDS_2	AGG	**GUG**	**UUA**	UGC	UAC
CDS_3	CGC	G**UG**	**CUU**	**UGC**	**U**AU
Amino acids of the protein	R	V	L	C	Y

Table 1: A possible solution with 3 CDSs. Example for the computation of the length of repeated or common substrings. $GUGUUA$ is the longest common substring between all pairs of CDSs, although there are other repeated substrings, e.g. in CDS_3, $UGCU$, but they have a lower length.

$$MLRCS = \max_{1 \leq i \leq j \leq N} \frac{LRCS(CDS_i, CDS_j)}{L}, \tag{5}$$

155 where L is the length in nucleotides of the CDSs. For every pair of CDSs, CDS_i and CDS_j (observe that $i = j$ is allowed), $LRCS$ is calculated as in Equation 6.

$$LRCS(CDS_i, CDS_j) = \underset{1 \leq p,q,l \leq L}{length(S_{i,p,l})} \quad when\ (S_{i,p,l} = S_{j,q,l}), \tag{6}$$

where L, as said, is the length of the CDSs, and if $p = q$ then $i \neq j$.

3. Multi-Objective Artificial Bee Colony (MOABC) Algorithm

160 Artificial Bee Colony (ABC) is an algorithm introduced by Dervis Karaboga in 2007 [20], motivated by the intelligent behavior of honey bees and used as an optimization tool.

ABC defines a set of operations in order to find pseudo-optimal solutions in a similar way to how bees find their best food sources. Individuals are members 165 of a colony where each type of bee has a task and there are three types of bees:

- Employed bees are associated to a specific food source (solution).

- Onlooker bees watch the dances of employed bees and choose the most profitable food sources depending on the dances.
- Scout bees carry out random searches for discovering new food sources.

In our approach, a Multi-Objective (MO) adaptation of ABC algorithm (MOABC) has been designed and implemented to achieve good solutions for our problem. In particular, as we address a multi-objective optimization problem, there is not one only best solution, but the result will be a set of trade-off solutions that optimize in different ways the considered objectives.

An important concept used in MO optimization problems is the Pareto dominance between two solutions. A solution x dominates another solution y (represented as $x \succ y$) when the values obtained by x in all the objective functions are always better than or equal to the corresponding values obtained by y, and at the same time, x obtains a better value in at least one of the objective functions. In the same way, we say that a solution is non-dominated or Pareto optimal if there is no other solution that dominates it. The graphical representation of the non-dominated solution set (or Pareto set) is known as Pareto front. The Pareto set represents the subset of best solutions found for the multi-objective optimization problem.

Remember that we represent a solution as a set of equal-length sequences of characters (see Table 1 for an example). Algorithm 1 shows the pseudo-code of the MOABC proposed. The first step is to establish a empty file to store non-dominated solutions. Then we initialize the colony with a total of *colony_size* solutions (line 2). Each individual in the colony is created randomly, except one, that is created by selecting each codon with the highest weight, and therefore, its $mCAI$ value will be 1. This particular solution is a non-dominated solution and could help future generations to achieve high CAI values.

Algorithm 1 MOABC pseudo-code.

```
Input: $colony\_size$ (number of individuals/solutions), $max\_cycles$ (maximum
    number of cycles/generations), $limit$ (abandonment criterion), and $P_m$ (mu-
    tation probability)
Output: $nondominated\_file$ (file with the non-dominated solutions)
 1: $nondominated\_file \leftarrow \emptyset$
 2: $init\_colony(colony\_size)$
 3: for $cycle \leftarrow 1, max\_cycles$ do
 4:     $send\_employed\_bees(colony\_size, P_m)$
 5:     $rank\_and\_crowding(colony\_size)$
 6:     $calculate\_probabilities(colony\_size)$
 7:     $send\_onlooker\_bees(colony\_size, P_m)$
 8:     $send\_scout\_bees(limit, cycle)$
 9:     $rank\_and\_crowding(2 * colony\_size)$
10:     $update\_nondominated\_solutions(nondominated\_file)$
11: end for
```

Next, a for loop starts, which includes operations that make the bee colony evolve for max_cycles cycles or generations. Each search cycle involves the management of employed bees, onlooker bees, and scout bees.

Employed bees are the first ones to perform (line 4). Every employed bee has an associated solution and a random mutation (among the different types of mutation) is applied to each individual for improving the quality. The mutation operator changes a specific part of a solution with the goal of optimizing some of the objective functions. The mutated solution will only be selected if it dominates the original solution. In this study, we use four types of mutation (as said, every time a solution is mutated, one of these mutations will be randomly selected and applied):

1. For each CDS, each codon is randomly replaced by another encoding (synonymous codon), with a probability of P_m. We can observe an example of this mutation in Figure 1.

2. For the CDS with the minimum CAI value, each codon is replaced by other encoding with greater weight (if several synonymous codons have higher weight, one of them is randomly selected), with a probability of P_m. If the codon has the encoding with the highest weight, then there is not replacement. An example of this mutation is shown in Figure 2.
3. For the pair of CDSs with the minimum HD, each codon is randomly replaced by another encoding, with a probability of P_m. Figure 3 presents an example for this mutation.
4. Codons that are within the longest length common substring are randomly replaced by another encoding, with a probability of P_m. Figure 4 illustrates an example of this mutation.

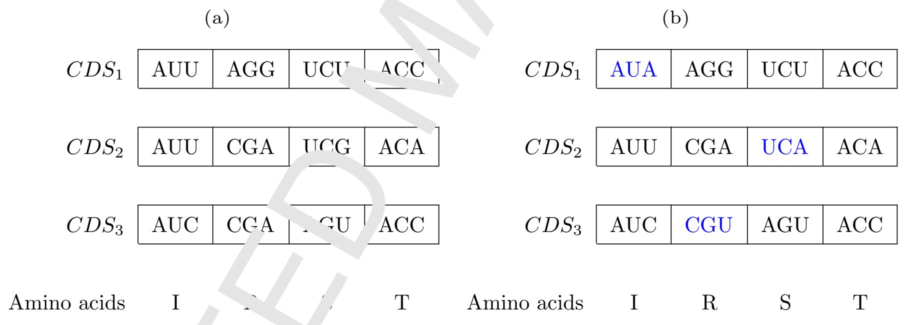


Figure 1: Example of random mutation in a solution with 3 CDSs, where (a) represents the original solution and (b) represents the mutated solution. After the mutation, the first codon in CDS_1, the third codon in CDS_2, and the second codon in CDS_3 are replaced by synonymous codons (in blue) randomly selected.

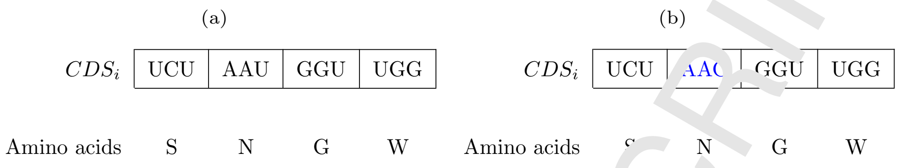


Figure 2: Example of mutation in CDS_i, which is the CDS with the lowest CAI value in a solution with n CDSs. In the original solution (a), the CAI value is 0.966. Applying the mutation (b), the second codon is replaced by another synonymous codon (in blue) with greater weight and, after that, CDS_i has a CAI value equal to 1.

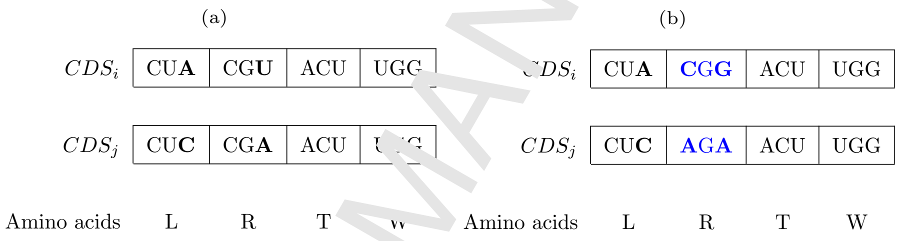


Figure 3: Example of mutation in CDS_i and CDS_j, which are the pair of CDSs with the minimum HD in a solution with n CDSs. In the original CDSs, shown in (a), only two nucleotides are different (in bold) between them, so the HD is 0.167. After the mutation (b), the second codon in each CDS has been replaced by another synonymous codon selected at random. Consequently, three nucleotides are different (in bold) and the HD has been incremented (0.250) between these CDSs.

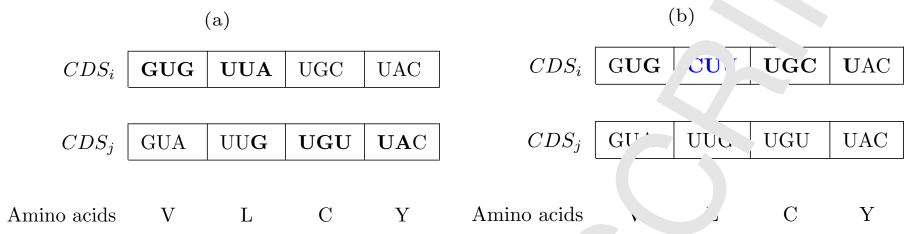


Figure 4: Example of mutation in a solution with n CDSs, where CDS_i and CDS_j are the pair of CDSs that contain the longest repeated or common substring. In the original CDSs shown in (a), the longest common substring is $GUGUUA$ (in bold). After the mutation (b), the second codon of CDS_i has been replaced by another synonymous codon (in blue) selected at random. Therefore, the previous common substring has been broken and the new longest common substring is $UGCU$ (in bold) in CDS_i, which is a shorter one.

In the onlooker bees phase (line 7) the colony size is duplicated. Every onlooker bee chooses an employed bee as its initial solution. This selection is based on the probability value associated (line 6) with each employed bee. Employed bees that represent better solutions will have higher probability values. To check which employed bees are better, they are evaluated and sorted by two metrics: rank and crowding (line 5). After a non-dominated sorting, the rank value indicates in which layer of the generated Pareto fronts a solution is. This non-dominated sorting is calculated taking into account the dominance relations of all the solutions. The second metric computes the solutions' crowding distance. The greater the crowding distance among the solutions, the greater the solutions diversity. A more detailed explanation of these two metrics can be found in [27].

Every onlooker bee will try to improve the selected employed bee (its initial solution) by using the same random mutation procedure explained above. In this case, the mutated solution is selected if it is non-dominated by the original one, otherwise the original solution is kept.

When the scout bees are processed (line 8), they check how many times a solution has been mutated without success (that is, without improving it). If this number of times exceeds an established *limit* it means that the solution will not be able to be improved, it has been exhausted, so it is abandoned and replaced with a new random solution. Furthermore, this new random solution is mutated n times, proportional to the current cycle, so that it can compete in the next cycle with the other solutions in the colony.

Before next cycle, the colony is reduced by half (its original size). For selecting the best solutions, these are ordered by rank and crowding again, with a difference: this process is applied to the whole colony, with double size of the original one (line 9). Finally, the non-dominated solutions in this cycle are stored in the *nondominated_file* (line 10) and then the next cycle can begin.

After explaining the MOABC pseudo code, we can highlight its strengths and advantages. It is a multi-objective evolutionary algorithm based on the intelligent behavior of a swarm of bees and their task social cooperation. From a theoretical perspective, swarm intelligence approaches are built upon the definition of autonomous agents with specific roles that cooperate together to achieve a common goal. These agents belong to different classes with different tasks and local rules, whose interactions lead to the attainment of global, collective intelligence through the sharing of information among all the components of the swarm.

In MOABC, the division of labor is implemented through the identification of different bee classes, each one with a well-defined role. While employed bees check the neighborhood of solutions previously identified by the swarm, onlooker bees exploit the most promising solutions found by the employed bees. On the other hand, scout bees have the responsibility of addressing local optima issues, identifying stagnant solutions and performing exploration tasks to find new promising candidates in unexplored search space regions. The sharing of information involves the communication of high-quality solutions from the employed bees to the onlooker ones, the identification of exhausted employed/onlooker solutions that must be replaced by the scout bees, and the definition of the next generation employed solutions from all the agents in the swarm. In this way, the population is processed by the local rules of each bee class, whose interactions allow the global spread of information among all the members thus leading to the global improvement of the population.

Please observe that, by considering different bee categories, the implemented mutation operator can be applied in different ways in accordance with the role (exploitation/exploration) of the current bee under processing. More specifically, while checking the neighborhood (employed) and exploiting high-quality solutions (onlooker), new solutions are generated by applying once the mutation operator. On the other hand, the exploration tasks (scout) involve multiple applications of the operator over randomly generated solutions.

In comparison to other evolutionary methods, MOABC undertakes the optimization process following the task division, interactions, and self-organization of the components of the bee swarm. The definition of employed, onlooker, and scout bees allows the integration of multiple search strategies (including exploitation and exploration-oriented procedures), which are applied in accordance with the current status of the population. In this sense, the inclusion of the *limit* control parameter allows the algorithm to quickly identify the presence of local optima, applying a specific technique (scout searches) to deal with them. The sharing of information between bees also represents a distinctive feature over traditional evolutionary designs, as it allows the search engine to address the problem by considering all the information gathered by the entire swarm. All these elements give rise to a robust search engine that can boost the solution of complex optimization problems like the design of CDSs. In fact, the literature gives account of the relevance of ABC in comparison to other approaches (such as genetic algorithms and differential evolution) in multiple sets of numerical test functions [28], multi-variable [20], and multi-dimensional [29] scenarios. Improved results have also been reported for the case of real-world problems [21], including hard-to-solve problems from the biological domain [30, 31].

In this study, this pseudo-code has been implemented in C/C++. The next section shows the data sets used in our experiments, the parameter settings for MOABC, the results obtained, the comparison with the results found in the literature, and the statistical analysis of the results.

4. Experiments and Results

As we have compared with the results from Terai et al. [19] in our experiments, we used nine proteins as a representative sample based on two attributes: length of the protein (in AA, Amino Acids) and the number of CDSs for that protein. Since both attributes influence the complexity of the instance, we balanced these two attributes. In particular, Table 2 shows how these nine proteins have different trade-offs between length and number of CDSs, so for a larger number of CDSs, we chose a protein with a smaller length and vice versa. Observe that this table includes instances with very different number of CDSs and lengths, and also, with different complexities, being a representative set of instances. We have used the Universal Protein Resource (UniProt [1]) to get the FASTA format for every protein.

Code	Name	CDSs	Length (AA)	CDSs*Length
Q5VZP5	DUS27_HUMAN	2	1158	2316
A4Y1B6	FADB_SHEPC	3	716	2148
B3LS90	OCA5_YEAS1	4	679	2716
B4TWR7	CAIT_SALSV	5	505	2525
Q91X51	GORS1_MOUSE	6	446	2676
Q89BP2	DAPE_BRADU	7	388	2716
A6L9J9	TRPF_PARD8	8	221	1768
Q88X33	Y1415_LACPL	9	114	1026
B7KHU9	PETG_CYAP7	10	38	380

Table 2: List of proteins used in the experiments.

For each instance in Table 2, we applied the algorithm explained in the previous section and we compared with the results obtained in the method implemented by Terai et al. [19] (based on NSGA-II algorithm). For getting

[1]`http://www.uniprot.org/uniprot/`

these results, we used the web-based application[2] provided by these authors, using their default parameters set.

In order to provide a fair comparison, the value of the parameters for the colony size and the number of generations were set to the same value as in the NSGA-II algorithm, that is, the colony size is 100 individuals (solutions) and the number of generations (*max_cycles*) is equal to 100. In MOABC algorithm, we adjusted other two parameters: the maximum number of attempts (or *limit*) to improve an employed or onlooker bee (10 attempts in our case) and the probability of mutation (P_m) that is equal to 0.05 (5%). We experimented with different values for these two parameters to find the best configuration. Specifically, Table 3 shows the values tested and the best adjustments are given by the highlighted values.

Furthermore, to ensure reliable statistics we repeated every experiment 31 times due to the stochastic nature of MOABC algorithm.

	Checked values						
P_m	1.25%	2.5%	**5%**	10%	20%	30%	40%
limit	2	5	**10**	15			

Table 3: All tested values to find the best configuration. The best value is highlighted in bold.

To evaluate the quality of the results, we made use of two indicators widely used in multi-objective optimization: hypervolume [32] and set coverage [33]. We also realized a statistical analysis to find the statistical significance and to be sure that the results are not likely to occur randomly.

4.1. Hypervolume indicator

The hypervolume indicator (HV), also known as Lebesgue measure [34], is a unary indicator of the quality of a set of non-dominated solutions. In the case of 3 objective functions, this measure computes the volume (in percentage) of

[2] `http://tandem.trahed.jp/tandem/`

the objective space covered by a Pareto front $\mathcal{A}$ with points $(a_1, a_2, \ldots, a_{|\mathcal{A}|})$, taking into account a reference point r. Equation 7 indicates how to calculate the value of HV.

$$HV(\mathcal{A}, r) = Leb\left(\bigcup_{i=1}^{|\mathcal{A}|} h(a_i, r)\right), \tag{7}$$

where *Leb* refers to the Lebesgue measure, $|\mathcal{A}|$ is the size (cardinality) of set $\mathcal{A}$, and $h(a_i, r)$ is the volume of the cube defined by each point in $\mathcal{A}$ (taking also into account the reference point).

Table 4 indicates the nadir and ideal values used in the hypervolume computations for all the proteins. These values were obtained by considering the results in all our experiments. Taking into account these values, the hypervolume calculations have been performed over objective scores normalized in the scale [0,1] to avoid the influence of different ranges in the objective values.

Objective	**Nadir value**	**Ideal value**
mCAI	0	1
mHD	0	0.35
MLRCS	1	0

Table 4: Nadir and ideal values used in the hypervolume computations and normalizations for all the proteins.

The HV indicator has been calculated by the same way for both algorithms. Table 5 shows median HV results and their quartile deviations calculated after assessing each protein individually. In almost all the proteins, the HV values obtained by the MOABC algorithm are better than those from the method proposed by Terai et al. [19], with the exception of the protein $Q5VZP5$ (as we will see, in this case, the differences between both algorithms are not statistically significant). Finally, the last row shows the average value among the nine proteins. Again, we can conclude that the MOABC algorithm is better than the NSGA-II algorithm [19]. This means that MOABC is able to obtain better

Pareto fronts, which include better non-dominated solutions that cover a higher volume of the objective space.

Protein	**MOABC**	**NSGA-II [19]**
Q5VZP5	$68.43\%_{\pm 0.62\%}$	$\mathbf{68.48\%}_{\pm 0.002[illegible]\%}$
A4Y1B6	$\mathbf{60.35\%}_{\pm 0.26\%}$	$60.05\%_{\pm 0.0007\%}$
B3LS90	$\mathbf{63.73\%}_{\pm 0.14\%}$	$62.[illegible]\%_{\pm 0.0018\%}$
B4TWR7	$\mathbf{57.06\%}_{\pm 0.15\%}$	$55.[illegible]\%_{\pm 0.0024\%}$
Q91X51	$\mathbf{59.71\%}_{\pm 0.19\%}$	$57.68\%_{\pm 0.0026\%}$
Q89BP2	$\mathbf{57.42\%}_{\pm 0.15\%}$	$55.55\%_{\pm 0.0025\%}$
A6L9J9	$\mathbf{53.77\%}_{\pm 0.16\%}$	$52.07\%_{\pm 0.0015\%}$
Q88X33	$\mathbf{48.71\%}_{\pm 0.2[illegible]\%}$	$46.93\%_{\pm 0.0010\%}$
B7KHU9	$\mathbf{47.71\%}_{\pm 0.54\%}$	$43.72\%_{\pm 0.0014\%}$
Average	**57.43%**	55.91%

Table 5: Results for hypervolume indicator, in the format: median$_{\pm quartile\_deviation}$. In bold we highlight the better results.

In addition, Figure 5 shows a visual representation for one of the proteins, $B7KHU9$. The objective values are already normalized in the scale [0,1] taking into account Table 4. As can be seen from that graph, the points in the projection (MLRCS, mHD) are near for both algorithms. By contrast, for the projections (mCAI, mHD) and (mCAI, MLRCS), in most of the points, the solutions from MOABC are better than the solutions from NSGA-II. Moreover, MOABC covers more regions of the objective space than NSGA-II. These improvements explain the clear advantage of MOABC with respect to NSGA-II. For the rest of the analyzed proteins, the behavior is similar.

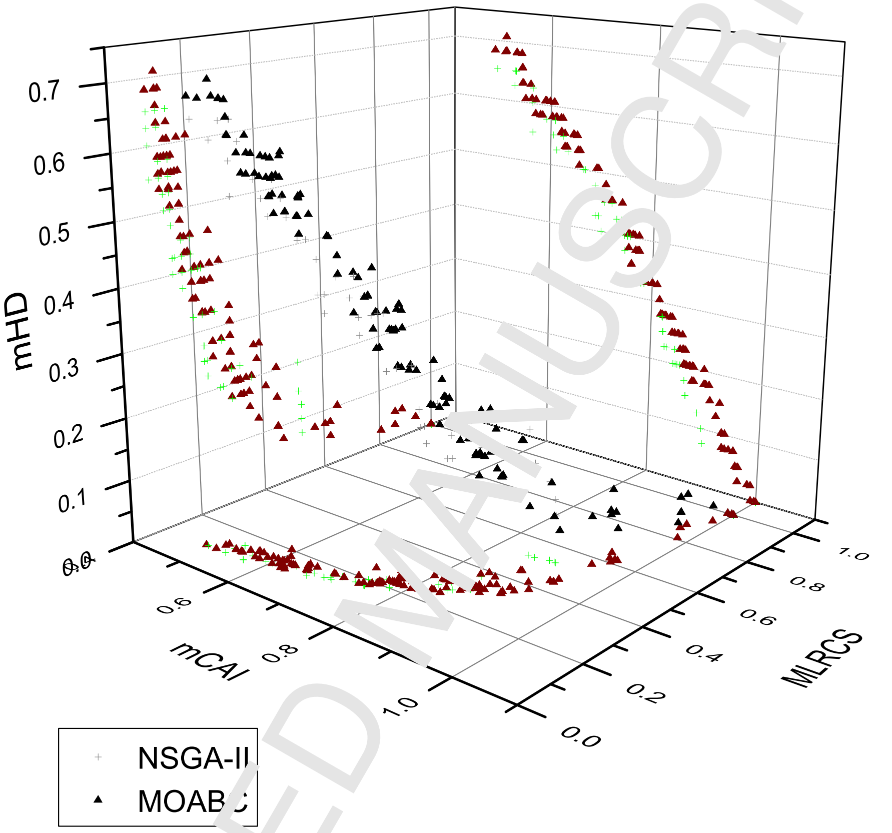


Figure 5: 3D scatter plot of the median Pareto fronts for the $B7KHU9$ protein. The points in the different 2D projections appear in red (MOABC) or green (NSGA-II), using the corresponding symbol.

4.2. Set Coverage indicator

Set coverage (SC) is the second quality indicator used. In this case, it is a binary-type indicator. This measure is based on how many solutions/points belonging to a Pareto front $\mathcal{B}$ are covered by solutions/points from another Pareto front $\mathcal{A}$. As Equation 8 shows, a solution b_j is said to be covered if there is a solution a_i that dominates to b_j or is equal to this one.

$$SC(\mathcal{A},\mathcal{B}) = \frac{|\{b_j \in \mathcal{B}; \exists\ a_i \in \mathcal{A} : a_i \succeq b_j\}|}{|\mathcal{B}|}, \tag{8}$$

where $|\mathcal{B}|$ is the size (cardinality) of set $\mathcal{B}$.

If each solution in $\mathcal{B}$ is covered at least by a solution in $\mathcal{A}$ then SC($\mathcal{A}$,$\mathcal{B}$) is equal to 1. Otherwise, if none of the solutions belonging to $\mathcal{B}$ is covered by the solutions in $\mathcal{A}$, SC($\mathcal{A}$,$\mathcal{B}$) will be 0. This weak-dominance operator is not symmetric and it can happen that $SC(\mathcal{B}, \mathcal{A}) \neq 1 - SC(\mathcal{A}, \mathcal{B})$. For this reason, we calculated the measure SC in both directions for each protein: SC(MOABC,NSGA-II) and SC(NSGA-II,MOABC).

In particular, Table 6 shows the results obtained and we can say that in almost all the cases, the MOABC algorithm attains improved set coverage scores than NSGA-II, except in the protein *B4TWR7*, where they have close values. This means that, for each instance, there are many more points of the Pareto front from MOABC algorithm that dominate points of the Pareto front from NSGA-II algorithm than vice versa. Again, in average, the solutions of MOABC algorithm are clearly better, covering an important percentage of the solutions obtained by the method proposed by Terai et al. [19].

Protein	**SC(MOABC, NSGA-II [19])**	**SC(NSGA-II [19], MOABC)**
Q5VZP5	**28.00%**	14.87%
A4YHB6	**54.00%**	4.26%
B3LS90	**34.00%**	14.62%
B4TWR7	23.00%	**24.86%**
Q9HK51	**28.00%**	6.63%
Q89BP2	**53.00%**	0.64%
A5L9J9	**36.00%**	5.43%
Q88X33	**54.00%**	0.62%
B7KHU9	**62.00%**	7.75%
Average	**41.33%**	8.85%

Table 6: Results for Set Coverage indicator. In bold we highlight the better results.

4.3. Statistical significance

In order to detect if there is a statistical significance in the results obtained, we performed a statistical analysis using a significance level (*p-value*) of 0.05 (5%) or confidence level of 95%. A detailed explanation of all the statistical tests used can be found in [35]. We try to apply a parametric analysis such as ANalysis Of VAriance (ANOVA) but before we should ensure that the samples follow a normal distribution and they have homogeneous variances. For this statistical study we use the results from the hypervolume indicator.

Firstly, the null hypothesis to be tested by the Kolmogorov-Smirnov test (KS-test) is that the sample has a normal distribution. As we can see in Table 7, none of the cases rejects this null hypothesis, therefore, we can say that all samples have a normal distribution with a confidence level of 95%.

Protein	**KS-test**		**Pass?**
	MOABC	**NSGAII [19]**	
Q5VZP5	0.122	0.200	Yes
A4Y1B6	0.200	0.200	Yes
B3LS90	0.0[illegible]3	0.147	Yes
B4TWR7	0.200	0.200	Yes
Q91X[illegible]	0.200	0.200	Yes
Q89BP2	0.200	0.200	Yes
A6L9J9	0.200	0.200	Yes
Q88X33	0.187	0.200	Yes
B7KHU9	0.180	0.200	Yes

Table 7: Normality analysis using Kolmogorov-Smirnov test.

The following test is Levene test to check if the two samples (from both algorithms) have homogeneous variances (null hypothesis). The results of this test for the 9 proteins are shown in Table 8. Three cases rejected this null hypothesis so this means that these ones cannot be tested by ANOVA and are analyzed by a nonparametric test such as the Mann-Whitney U test. All other cases are tested by ANOVA.

Protein	**Levene test**	**ANOVA**	**Mann-Whitney U test**	**Statistical significance**
Q5VZP5	0.149	0.907	–	No
A4Y1B6	0.020	–	0.111	No
B3LS90	0.457	0.000	–	Yes
B4TWR7	0.313	0.000	–	Yes
Q91X51	0.147	0.000	–	Yes
Q89BP2	0.019	–	0.000	Yes
A6L9J9	0.400	0.000	–	Yes
Q88X33	0.068	0.000	–	Yes
B7KHU9	0.015	–	0.000	Yes

Table 8: Results of the Levene test to check the homogeneous variances and results of the ANOVA or the Mann-Whitney U test depending on the previous tests to find statistical significance.

405 Thus, the null hypothesis to be tested by ANOVA and the Mann-Whitney U test is that both samples of the same experiment are similar and there are no statistically significant differences between them. As Table 8 shows, seven of nine experiments reject this null hypothesis, therefore, we can say that these seven experiments have a statistical significance with a confidence interval of 410 95%. By contrast two of the experiments, $Q5VZP5$ and $A4Y1B6$, accept the null hypothesis and this implies that there are no statistically significant differences between the tested samples.

All in all, these results match the hypervolume values shown in Table 5. In overall terms, our method accomplishes a more satisfying behavior than NSGA-415 II, attaining statistically significant improvements in most of the evaluation scenarios. Particularly, the MOABC algorithm achieves Pareto fronts (solutions) different and better than the NSGA-II algorithm [19] in 7 out of 9 instances.

5. Conclusion and Future Work

In this work, we have proposed a novel method for the multi-objective design of CDSs encoding the same protein, which is an important task in bioinformatics (and more specifically, in synthetic biology). The tackled multiobjective problem involves three fundamental objective functions (CAI, HD, LRCS). In our proposal, we have adapted the ABC algorithm to the multi-objective context and we have designed and implemented the MOABC (Multi-Objective Artificial Bee Colony) algorithm for doing this task. In conclusion, we propose to use the MOABC algorithm for achieving a range of solutions that best encode a protein with several CDSs, taking into account that the nucleotide sequences should be as different as possible (between both different CDSs and different subsequences within the same CDS, thus avoiding the homologous recombination) and at the same time the codon adaptation indexes should be as high as possible. To get solutions that allow us to make fair comparisons with other techniques from the literature (NSGA-II), we used the same codon usage frequencies, and both methods used the same colony/population size and number of generations. The experiments have been done over 9 real protein instances. These instances combine different lengths and number of CDSs, therefore, being a representative set of instances. The results show that MOABC achieves better Pareto fronts (solutions) than the results previously published in the literature in almost all the instances, being not statistically significant the differences in the only instance where MOABC obtained little bit worse results.

As future work, we have planned to use other alternative multi-objective algorithms to compare them with MOABC, allowing us to further evaluate the good quality of the MOABC results or even improving these results. Moreover, the number of existing MOABC algorithms is high (e.g. [22, 23, 24, 25]), therefore, the comparison of several of them in the problem under study is of interest for a next research work. This will imply their design for this specific problem, their implementation, their execution, and finally, their comparison. On the other hand, due to the good results obtained by MOABC, we intend to apply this multi-objective algorithm to other bioinformatics multi-objective problems, assessing if MOABC also obtains good results in these other problems.

Acknowledgements

This work was partially funded by the AEI (State Research Agency, Spain) and the ERDF (European Regional Development Fund, EU), under the contract TIN2016-76259-P (PROTEIN project). Thanks also to the Junta de Extremadura and the ERDF for the grant GR18090 provided to the research group TIC015. Sergio Santander-Jiménez is supported by the Post-Doctoral Fellowship from FCT (Fundação para a Ciência e a Tecnologia, Portugal) under grant SFRH/BPD/119220/2016.

References

[1] A. Vassileva, D. A. Chugh, S. Swaminathan, N. Khanna, Expression of hepatitis B surface antigen in the methylotrophic yeast Pichia pastoris using the GAP promoter, Journal of Biotechnology 88 (1) (2001) 21–35. `doi:10.1016/S0168-1656(01)00254-1`.

[2] H. Hohenblum, B. Gasser, M. Maurer, N. Borth, D. Mattanovich, Effects of gene dosage, promoters, and substrates on unfolded protein stress of recombinant Pichia pastoris, Biotechnology and Bioengineering 85 (4) (2004) 367–375. `doi:10.1002/bit.10904`.

[3] P. Gu, F. Yang, T. Su, Q. Wang, Q. Liang, Q. Qi, A rapid and reliable strategy for chromosomal integration of gene(s) with multiple copies, Scientific Reports 5 (2015) 9684. `doi:10.1038/srep09684`.

[4] K. E. J. Tyo, P. K. Ajikumar, G. Stephanopoulos, Stabilized gene duplication enables long-term selection-free heterologous pathway expression, Nature Biotechnology 27 (2009) 760–765. doi:10.1038/nbt.1555.

[5] C. A. Scorer, J. J. Clare, W. R. McCombie, M. A. Romanos, K. Sreekrishna, Rapid selection using G418 of high copy number transformants of Pichia pastoris for high-level foreign gene expression, Bio/Technology 12 (1994) 181–184. doi:10.1038/nbt0294-181.

[6] R. Aw, K. M. Polizzi, Can too many copies spoil the broth?, Microbial Cell Factories 12 (1) (2013) 128. doi:10.1186/1475-2859-12-128.

[7] P. Shen, H. V. Huang, Homologous recombination in Escherichia coli: dependence on substrate length and homology, Genetics 112 (3) (1986) 441–457.

[8] F. K. Khasanov, D. J. Zvingila, A. A. Zainullin, A. A. Prozorov, V. I. Bashkirov, Homologous recombination between plasmid and chromosomal DNA in Bacillus subtilis requires approximately 70 bp of homology, Molecular and General Genetics 234 (3) (1992) 494–497. doi:10.1007/BF00538711.

[9] P. Manivasakam, S. C. Weber, J. McElver, R. H. Schiestl, Micro-homology mediated PCR targeting in Saccharomyces cerevisiae, Nucleic Acids Research 23 (14) (1995) 2799–2800. doi:10.1093/nar/23.14.2799.

[10] J. Athey, A. Alexaki, E. Osipova, A. Rostovtsev, L. V. Santana-Quintero, U. Katneni, V. Simonyan, C. Kimchi-Sarfaty, A new and updated resource for codon usage tables, BMC Bioinformatics 18 (1) (2017) 391. doi:10.1186/s12859-017-1793-7.

[11] G. R. Webster, A. Y.-H. Teh, J. K.-C. Ma, Synthetic gene design - The rationale for codon optimization and implications for molecular pharming in plants, Biotechnology and Bioengineering 114 (3) (2017) 492–502. doi:10.1002/bit.26183.

[12] C.-H. Yu, Y. Dang, Z. Zhou, C. Wu, F. Zhao, M. S. Sachs, Y. Liu, Codon usage influences the local rate of translation elongation to regulate co-translational protein folding, Molecular Cell 59 (5) (2015) 744–754. doi:10.1016/j.molcel.2015.07.018.

[13] T.-A. Tran, N. T. Vo, H. D. Nguyen, B. T. Pham, A novel method to predict highly expressed genes based on radius clustering and relative synonymous codon usage, Journal of Computational Biology 22 (12) (2015) 1086–1096. doi:10.1089/cmb.2015.0121.

[14] J. X. Chin, B. K.-S. Chung, D.-Y. Lee, Codon Optimization OnLine (COOL): a web-based multi-objective optimization platform for synthetic gene design, Bioinformatics 30 (15) (2014) 2210–2212. doi:10.1093/bioinformatics/btu192.

[15] J. C. Guimaraes, M. Rocha, A. P. Arkin, G. Cambray, D-Tailor: automated analysis and design of DNA sequences, Bioinformatics 30 (8) (2014) 1087–1094. doi:10.1093/bioinformatics/btt742.

[16] P. Puigbò, E. Guzmán, A. Romeu, S. Garcia-Vallvé, OPTIMIZER: a web server for optimizing the codon usage of DNA sequences, Nucleic Acids Research 35 (suppl_2) (2007) W126–W131. doi:10.1093/nar/gkm219.

[17] I. Frumkin, M. J. Lajoie, C. J. Gregg, G. Hornung, G. M. Church, Y. Pilpel, Codon usage of highly expressed genes affects proteome-wide translation efficiency, Proceedings of the National Academy of Sciences 115 (21) (2018) E4940–E4949. doi:10.1073/pnas.1719375115.

[18] A. Uddin, S. Chakraborty, Codon usage pattern of genes involved in central nervous system, Molecular Neurobiology (2018) 1–12.doi:10.1007/s12035-018-1173-y.

[19] G. Terai, S. Kamegai, A. Taneda, K. Asai, Evolutionary design of multiple genes encoding the same protein, Bioinformatics 33 (11) (2017) 1613–1620. doi:10.1093/bioinformatics/btx030.

[20] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, Journal of Global Optimization 39 (3) (2007) 459–471. doi:10.1007/s10898-007-9149-x.

[21] D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, A comprehensive survey: artificial bee colony (ABC) algorithm and applications, Artificial Intelligence Review 42 (1) (2014) 21–57. doi:10.1007/s10462-012-9328-0.

[22] R. Akbari, R. Hedayatzadeh, K. Ziarati, B. Hassanizadeh, A multi-objective artificial bee colony algorithm, Swarm and Evolutionary Computation 2 (2012) 39–52. doi:10.1016/j.swevo.2011.08.001.

[23] Y. Huo, Y. Zhuang, J. Gu, S. Ni, Elite-guided multi-objective artificial bee colony algorithm, Applied Soft Computing 32 (2015) 199–210. doi:10.1016/j.asoc.2015.03.040.

[24] M. Ding, H. Chen, N. Lin, S. Jing, F. Liu, X. Liang, W. Liu, Dynamic population artificial bee colony algorithm for multi-objective optimal power flow, Saudi Journal of Biological Sciences 24 (3) (2017) 703–710. doi:10.1016/j.sjbs.2017.01.045.

[25] E. Hancer, B. Xue, M. Zhang, D. Karaboga, B. Akay, Pareto front feature selection based on artificial bee colony optimization, Information Sciences 422 (2018) 462–479. doi:10.1016/j.ins.2017.09.028.

[26] P. M. Sharp, W.-H. Li, The codon adaptation index - A measure of directional synonymous codon usage bias, and its potential applications, Nucleic Acids Research 15 (3) (1987) 1281–1295. doi:10.1093/nar/15.3.1281.

[27] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197. doi:10.1109/4235.996017.

[28] D. Karaboga, B. Akay, A comparative study of artificial bee colony algorithm, Applied Mathematics and Computation 214 (1) (2009) 108–132. doi:10.1016/j.amc.2009.03.090.

[29] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Applied Soft Computing 8 (1) (2008) 687–697. doi:10.1016/j.asoc.2007.05.007.

[30] X. Lei, J. Sun, X. Xu, L. Guo, Artificial bee colony algorithm for solving multiple sequence alignment, in: IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA), IEEE, 2010, pp. 337–342. doi:10.1109/BICTA.2010.5645304.

[31] B. A. Garro, R. A. Vazquez, K. Rodriguez, Classification of DNA microarrays using artificial bee colony (ABC) algorithm, in: International Conference in Swarm Intelligence, Advances in Swarm Intelligence, Vol. 8794 of LNCS, Springer, 2014, pp. 207–214. doi:10.1007/978-3-319-11857-4_24.

[32] N. Beume, C. M. Fonseca, M. Lopez-Ibanez, L. Paquete, J. Vahrenhold, On the complexity of computing the hypervolume indicator, IEEE Transactions on Evolutionary Computation 13 (5) (2009) 1075–1082. doi:10.1109/tevc.2009.2015575.

[33] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, V. G. da Fonseca, Performance assessment of multiobjective optimizers: an analysis and review, IEEE Transactions on Evolutionary Computation 7 (2) (2003) 117–132. doi:10.1109/tevc.2003.810758.

[34] R. G. Bartle, The elements of integration and Lebesgue measure, John Wiley & Sons, Inc., 1995. doi:10.1002/9781118164471.

[35] D. J. Sheskin, Handbook of parametric and nonparametric statistical procedures, 5th Edition, Chapman & Hall/CRC, NY, USA, 2011.

Highlights (for review)

Highlights

- The design of multiple genes encoding the same protein is an important task

- This task can be tackled as a multi-objective optimization problem with 3 objectives

- We have designed and implemented a solution procedure based on the MOABC algorithm

- The experiments have been done over 9 real protein instances

- MOABC obtains better results than the ones found in the literature