Proceedings of ICTAMI
International Conference on Theory and Applications in Mathematics and Informatics
17th–20th of September, 2015, Alba-Iulia, Romania
Computer Science Section

pp. 179-194

DIRECTED EVOLUTION - A BIO-INSPIRED OPTIMIZATION TECHNIQUE

C. Rotar

Abstract. In recent decades we have witnessed an infusion of calculating models based on models offered by nature, models with more or less fidelity to the original that have led to the development of various problem-solving computational procedures. Technological progress today allows the accelerated reproduction of natural phenomena in the laboratory, which is why a new niche has arisen in the landscape of nature-inspired methods. This niche is devoted to the emulation of artificial biological processes in computational problem-solving methods. This paper proposes a novel approach, which is to develop computational methods in the field of Natural Computing based on semi-natural processes. In the first step we explain Directed Evolution, defined as the artificial reproduction of the process of evolution in the laboratory in order to obtain performing biological entities. For the computer scientist, this provides a strong source of inspiration in the search for efficient methods of optimization. The computational model we propose here largely overlaps with the Directed Evolution protocol, and the results obtained in the numerical experiments confirm the viability of such techniques inspired by processes which are more artificial than natural.

2010 *Mathematics Subject Classification*: 30C45.

Keywords: optimization, directed evolution, bio-inspired computing.

1. Introduction

Natural Computing is a significant branch of artificial intelligence research, and comprises a series of paradigms and computational techniques broadly inspired by nature. A detailed account of the field is given in [17], where Natural Computation is defined as the combination of three major research directions: *computing inspired by nature*, simulation and emulation of nature in computers, and computing with natural materials. Among these directions, computing inspired by nature seems to be the most popular, presenting different computational methods which have been successfully applied in solving various problems. Generally, the computing approach inspired by nature mimics observable phenomena in nature and borrows concepts and terminology from disciplines cognate to biology. Thanks to technological advances, phenomena are now produced in the laboratory. All these may be seen as powerful sources of inspiration in the development of computational techniques. We refer here to Directed Evolution, defined as an artificial process by which the process of evolution is recreated in the laboratory. Obviously, emulation of evolution, seen from the biological perspective, in the paradigm of artificial evolution, needs to be adjusted: the time required is reduced by speeding up the "evolution" process, and the process itself is guided and strictly controlled in order to obtain what is desired. Directed Evolution, viewed as a tool inspired by nature, seems to be a source worthy of consideration in the design of evolutionary algorithms, since, unlike natural evolution, it has the advantages of control and time.

The development of algorithms inspired by Directed Evolution would not fit any of the categories covered by Natural Computation. The strict plan of organization proposed in [17], which lists the three subfields of Natural Computing, would perhaps require the addition of a new branch, which might involve the infusion of one of the artificial technologies (synthetic biology) into the computational technique. The richness of the process, and the similarity with the evolutionary metaphor, thus led us to the idea of designing a computational model based on Directed Evolution. The proposal to formulate a new computational paradigm that is inspired by the artificial process of Directed Evolution represents an isolated and challenging enterprise in the landscape of bio-inspired techniques. Nevertheless, the richness of such a source of inspiration is detected in [4], which suggests the use of Directed Evolution techniques for solving the Hamiltonian Path problem. In [5], the Directed Evolution expresses evolutionary strategies that are accompanied with ingenious mechanisms for controlled mutation, and does not refer to the simulation of evolution in the laboratory. However, it does prefigure the need for *directing* the evolution in terms of obtaining better performances of the computational methods. The emergent nature of Directed Evolution is anticipated in [6], but the strong divergence between evolutionary computational tools and the research of molecular biologists is also highlighted.

Analysing the two branches of the different fields Directed Evolution in biology and evolutionary algorithms in computer science it may be noticed that the infusion from biology toward computation is inferior to the influence flowing in the opposite direction. Computational techniques represent more or less efficient methods for sustaining or optimizing Directed Evolution processes [7]. Moreover, the evolutionary computation techniques are applied for optimizing the process of Directed Evolution or else they are exploited as helpful tools [8], [9].

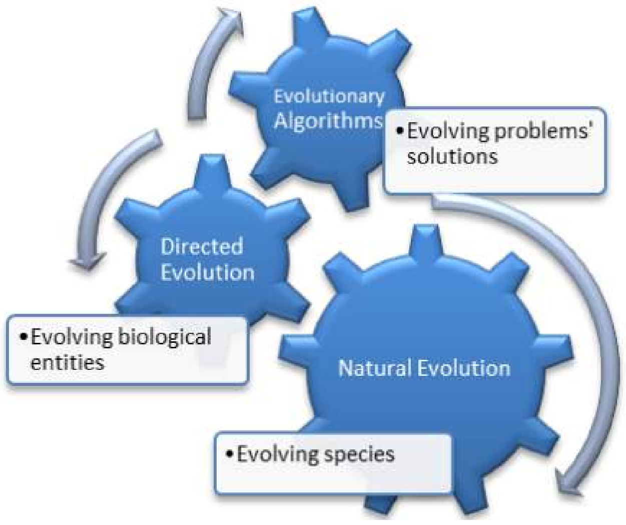

Figure 1: Synthetic description of the relations between paradigms

2. The Metaphor of Directed Evolution

The Directed Evolution [10] refers to a collection of procedures that are performed in the laboratory and by which evolution is simulated, aiming to generate molecules that cannot be found in nature. As a method, Directed Evolution involves several rounds, comprising diversification, amplification and selection of a large library of variants. Through these procedures, the beneficial mutations accumulate in the genetic pool, and scientists can thereby guide the evolution of biological entities towards the desired goal.

Directed Evolution mimics natural evolutionary process, but unlike it, it occurs at the molecular level, and does not create new organisms but only accentuates or produces new genetic traits. The process of Directed Evolution is possible due to research that was initiated by the enunciation of the Central Dogma (Crick, 1951), under which the transfer of biological information is mostly done in the following direction: DNA can be copied to DNA (DNA replication), DNA information can be copied into RNA (transcription), and proteins can be synthesized using the information in RNA as a template (translation). In short, directed evolution at the protein level can be defined as the evolving of proteins toward a user-defined goal, and it is an iterative process that involves the generation of a set of biological entities of interest (gene variants), and the screening/selection to identify those variants which display better properties. The best mutants of each iteration will serve as templates

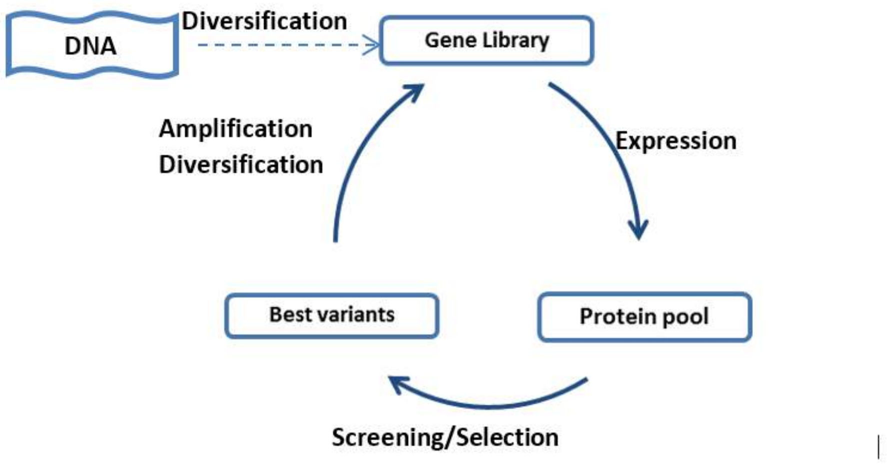

Figure 2: Schematic overview of directed evolution cycle.

for subsequent iterations of diversification and selection. The process is repeated until the desired improvement is achieved. This powerful engineering tool is an iterative strategy which includes at each round three major steps: amplification, diversification and selection [1], [2] [3].

Directed Evolution begins with a first phase whose outcome is a large library of genes. Specific tools of the first stage are random mutagenesis and gene recombination. The most widely used method of random mutagenesis is error-prone polymerase chain reaction (PCR) [11] which introduce mutations into the DNA chain. Genetic recombination, considered as the sexual component of diversification, involves the recombination of different genetic sequences in order to create new structures. One of the most robust techniques developed in this direction is DNA shuffling [12], which consists in the recombination of homologous genes. Directed Evolution involves the coupling of the genetic information stored in DNA or RNA with the functional information from proteins [15]. The proteins that are expressed by the produced library of genes require linkage to the correspondent genetic code [14], as long as the purpose of the screening or the selection process that takes place at the level of phenotype (the expressed proteins pool) is to identify and isolate the genetic signature (genotype) of those proteins with the desired features.

Subsequent to the diversification phase, the obtained molecules (proteins) are made the subject of the selection/screening in order to isolate the improved entities. The major difference between the two mechanisms of selection and screening is that selection is understood as a method of identifying the best variants by simul-

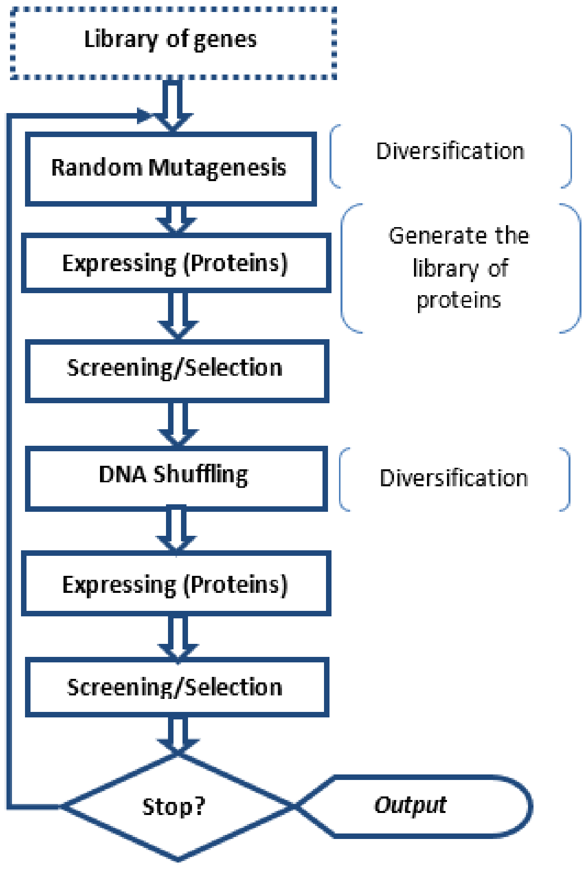

Figure 3: Schematic overview of a computational directed evolution algorithm.

taneously analysing the entire library, while screening is understood as a method of examination of each member of the library [2]. Next, the selected entities are subject to the amplification process (e.g., PCR), the process by which multiple copies of a gene or DNA sequence are created. During this process, the genetic information is diversified by the rare introduction of errors, cross-overs, and reorganizations [13]. The entire procedure repeats until the goal is achieved for example, obtaining a protein with specific functionality.

Next, a scheme of *Directed Evolution for problem solving* is described. The system contains: **structures** of numerical information which codify the possible solutions of the search space and by which the entities from biological model are represented (DNA, Proteins) and **computing procedures** which describe the corresponding processes from Directed Evolution (e.g. Diversification, Expression, Screening/Selection). The main elements (structures and modules) of the model described in Figure 3 are detailed next:

Modules:

- Diversification represents the process by which the gene library (analogous to the population from the genetic algorithm paradigm) is created. The specific procedures of diversification consist in the introduction of the genetic mutation and in the shuffling of the genetic code.
- Expression represents the process through which the partial solutions that are codified in genes are further evaluated conforming to the problems objectives. The outcome of this procedure is the protein library, corresponding to the objectives space.
- Screening/Selection mimics the corresponding screening and selection process from the natural paradigm, aiming to identify those elements which respond to the problems objectives by establishing correspondence between the genes and the expressed proteins.
- Amplification is the process by which the results obtained by selection or screening are amplified in order to obtain a new diverse library of genes. The procedures involved in this stage are the cloning of the genes and diversification. Amplification and Diversification are complementary and they are not distinct modules in the general algorithm.

3. Directed Evolution Algorithm as Optimization Tool

Let us consider a general optimization problem with m objectives and n variables. For the purpose of current research, we consider only optimization problems without constraints. The following paragraphs describe the structures and procedures involved in the Directed Evolution Algorithm.

3.1. Codification - structures

In a natural model, each gene is a sequence of nucleotides which form the DNA. In an artificial model, a gene corresponds to the possible solution from the search space and is represented by an n-dimensional vector of values from the specific alphabet: $(x_1, x_2, \dots, x_n)$ (Figure 4).

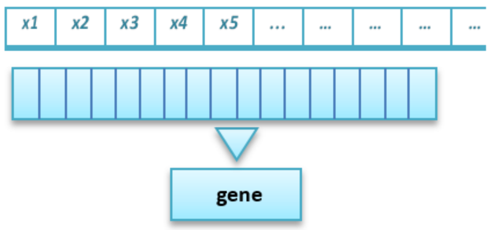

Figure 4: Gene structure

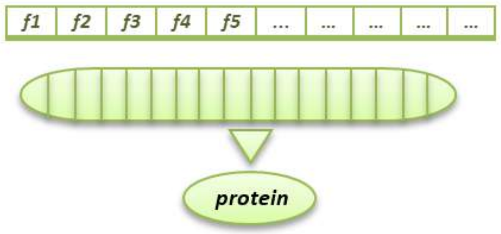

Figure 5: Protein structure

The natural protein represents a chain of aminoacids. The corresponding objectives values computed on the basis of the genes codification form the protein structures $(f_1, f_2, \ldots, f_m)$, which correspond to the m-dimensional vector as in Figure 5.

3.2. Libraries and Expressing

The algorithms work with two libraries: the first corresponds to the genes; the second corresponds to the expressed proteins. Both libraries vary in size, $S_{current} \in [S_{min}, S_{max}]$, during the directed evolution process.

The real proteins are generated through the process of gene expression in two stages: transcription and translation. Natural transcription is the process by which the genes are copied into the RNA from where, in the next phase, namely translation, the proteins are created. For the purpose of our study we simplified the expressing procedure, considering that the output of the transcription and translation is given by the proteome (the entire set of expressed proteins [29]). For those optimization problems that have constraints, the distinct phase of transcription would be suitable as it would allow the generation of a subset of the proteome, corresponding to the feasible solutions.

The procedure of Artificial Expressing represents the process by which the proteins are generated, respectively, the objectives values are computed.

The fitness (the quality of the gene giving the functionality of a protein) of a gene $G \in$ DNA is given by the quality of the corresponding expressed protein $P = Translate(G)$:

$$Fitness(G) = Quality(P) = Quality(Translaet(G))$$

where the function to evaluate the quality of the protein is designed according to the specification of the problem.

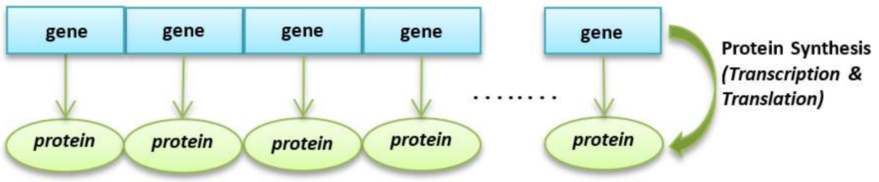

Figure 6: Gene expression. Protein synthesis. (For optimization problems without constrains).

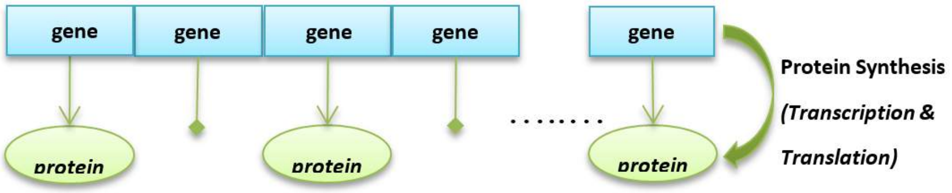

Figure 7: Gene expression. Protein synthesis. (For optimization problems involving constrains).

```
Function Translate ($G_i$, Protein$_i$)
for each objective j=1 to m
    Protein$_i$(j) ← $f_j(G_i)$ //compute the jth objectives
end for
End
```

```
Procedure Expressing
for each gene $G_i \in$ DNA
    Protein$_i$ ←Translate($G_i$) //protein expressing
end for
End
```

3.3. Selection/Screening

Selection and screening of the qualified sequences is conducted in order to spread the performant DNA variants. The decision by which the genetic sequences (obtained either by shuffling or by mutagenesis) survive is made on the basis of the qualities of the proteins, which are given by the values of the objectives codified in the proteins structures. The selection mechanism will favour the best candidates from the genetic library. The selection procedure collects the elite among the genetic pool according to the objective values. The elite acts for the next round of variation, increasing the chance of an overall improvement of the genetic library, and thus implicitly of the expressed proteins. In respect to the ratio between the initial size and the maximum size of the genetic library, we choose the same proportion of elite size from the current library (e.g., 20%). Simply speaking, the selection procedure chooses the set of the best solutions from the current library representing the gene pool for a next round of diversification.

```
Procedure Selection
Input DNA_new
    DNA← Elite(DNA_New)
Output DNA
End
```

3.4. Diversification: Mutagenesis and DNA Shuffling

Considering the genes' library DNA = $\{G_1, \ldots, G_{Scurrent}\}$ and $\prod = \{Protein_1, \ldots, Protein_{Scurrent}\}$ proteins library, the following procedures describes the major variation phases of the Directed Evolution Optimization Algorithm, namely DNA Shuffling and Mutagenesis. Diversification of the genetic code is made according to the DE protocol: mutagenesis and DNA Shuffling.

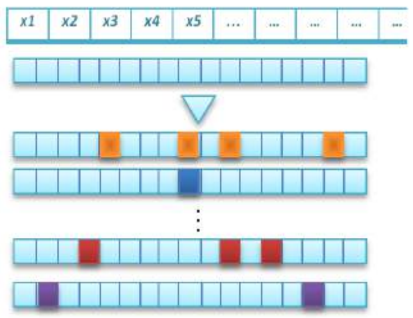

Figure 8: Mutagenesis. Mutated DNA sequence may vary differently

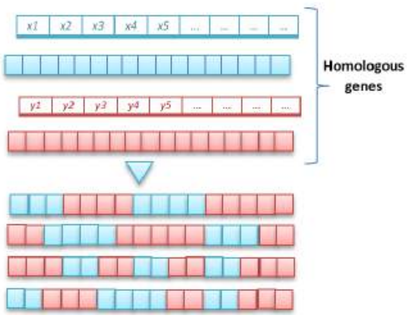

Figure 9: Shuffling of homologous genes

The *frequency* represents the average number of mutations per each gene and is computed according to the following formula:

$$frequency = S_{max}/S_{current}$$

where S_{max} is the maximum size of the library and $S_{current}$ represents the current size of the library. This value measures to what extent each variant proliferates into the genetic pool. The frequency is a constant value during the evolutionary process. Each new variant is generated within the boundaries of the gene library, as in the natural model where mutants are similar to the originals. Therefore, the range of the search space updates accordingly to the current library. The selected genetic sequence is mutated by randomly replacing a value (nucleotide) with a new one that is generated into the current range.

The DNA_Shuffling algorithm to some extent mimics the genetic shuffling procedure in Directed Evolution. Mating is made on the basis of the similarities of two genetic sequences. The similarity of two genes is computed through a distance measure. In our experiments we used either the Euclidian distance when genes are real values, or the Hamming distance when the binary alphabet is used. The homologous partner of one gene is the closest gene from the same library according to the distance measure. Put simply, the genetic shuffling procedure blends two homologous genes.

```
Procedure Mutagenesis
Input DNA –current genetic library
    for each gene $G_i \in DNA$
        for k=1 to frequency
            M← Mutate ($G_i$)
            Copy( $G\_new_i$ ,M)
        end for
    end for
DNA← {$G\_new_1$, …, $G\_new_{Smax}$}
Output DNA //new genetic library
End Mutagenesis
```

```
Procedure DNA_Shuffling
Input DNA –current genetic library
 for k=1 to $S_{current}$
        Copy( $G\_new_k$ ,$G_k$)
 end for
for k=$S_{current}$+1 to $S_{max}$
 Randomly select $G_i \in DNA$
  $H_i$← Homologous ($G_i$) , $H_i \in DNA$
  New← GeneticShuffle ($G_i$,$H_i$)
  Copy( $G\_new_i$ ,New)
end for
DNA← {$G\_new_1$, …, $G\_new_{Smax}$}
Output DNA // new genetic library
End DNA_Shuffling
```

The general Directed Evolution Algorithm for Optimization is an iterative technique with the following modules: diversification $(DNA_{Shuffling}, Mutagenesis)$, *Expression* and *Selection*.

```
Algorithm DirectedEvolution
Randomly generate DNA library of size $S_{min}$
 while (termination_condition*)
        Call Mutagenesis
        Call Expression
        Call Selection
        Call DNA_Shuffling
        Call Expression
        Call Selection
end while
EndAlgorithm
```

**termination_condition may refer to the attaining the pre-established maximum cycles*

4. Results and discussion

In order to assess the performance of the DE algorithm, we investigated four popular test functions (see for detailed description [19]) which are scalable as regards the number of variables. The algorithm runs 30 times and the Accuracy metric is computed. Accuracy measures the Euclidian distance between the best found solution and the global optimum. The number of function evaluations (NFE) is also recorded. We considered three test scenarios to observe the algorithms performance when the library size, the number of cycles and the number of variables varies:

1. **First scenario**: the maximum library size varies as 50, 100, 150, 200, 250, and 300 when the initial library size represents 20% of the maximum size. At each run a number of 100 cycles are produced and the number of variables is set to 30.

2. **Second scenario**: the cycles number varies as 50, 100, 200, 300, 400, 500. The maximum size of the library is 50 (initial size is 20

3. **Third scenario**: The number of variables varies (5, 10, 15, 20, 25, 30) . The maximum library size is 100 and 100 cycles are produced for each run.

Table **1** First test scenario. Results when library size varies.

SCENARIO I (library size varies, *const no. of iterations=100, const no. of variables=30)*
Average ACCURACY and NFE for 30 runs

Library Size	Metric	F1 Ackley		F2 Griewank		F3 Rastrigin		F4 Schwefel	
		Average	Std	Average	Std	Average	Std	Average	Std
50	*Acc*	2.69E-02	1.36E-02	2.64E-01	3.12E-01	6.16E-01	1.01E+00	6.28E+00	1.62E+01
	NFE	9.85E+03	1.08E+01	9.86E+03	8.58E+00	9.85E+03	1.06E+01	9.85E+03	8.58E+00
100	*Acc*	1.73E-02	5.46E-03	2.93E-01	2.95E-01	2.97E-02	2.44E-02	1.39E-01	1.19E-01
	NFE	1.97E+04	2.31E+01	1.97E+04	2.89E+01	1.97E+04	2.49E+01	1.97E+04	2.37E+01
150	*Acc*	1.27E-02	4.03E-03	1.65E-01	2.34E-01	1.42E-02	1.15E-02	7.74E-02	1.00E-01
	NFE	2.95E+04	4.02E+01	2.95E+04	3.49E+01	2.95E+04	3.67E+01	2.95E+04	3.81E+01
200	*Acc*	1.11E-02	3.79E-03	2.15E-01	2.58E-01	9.53E-03	6.54E-03	4.83E-02	4.78E-02
	NFE	3.93E+04	6.81E+01	3.93E+04	5.21E+01	3.93E+04	4.32E+01	3.93E+04	3.44E+01
250	*Acc*	1.00E-02	2.74E-03	1.74E-01	2.75E-01	6.23E-03	5.11E-03	3.80E-02	2.83E-02
	NFE	4.91E+04	6.15E+01	4.91E+04	6.96E+01	4.91E+04	5.61E+01	4.91E+04	6.47E+01
300	*Acc*	9.59E-03	3.35E-03	1.66E-01	2.63E-01	5.46E-03	3.02E-03	2.54E-02	1.49E-02
	NFE	5.89E+04	7.14E+01	5.89E+04	6.73E+01	5.89E+04	8.38E+01	5.89E+04	8.86E+01

As we expected, the accuracy of the solutions found by the DE algorithm becomes better with increasing size of the library. Exept for the Griewank test function, which is paricularly difficult as it is highly multimodal, for the other test functions the performance of the algorithm increases as the library size becomes larger.

Table **2** Second test scenario. Results when the number of iterations varies.

SCENARIO II (no of iteration variation, *library size =50, const no. of variables=30)* ***Average ACCURACY and NFE for 30 runs***									
Iteration No.	Metric	F1 Ackley		F2 Griewank		F3 Rastrigin			
		Average	Std	Average	Std	Average	Std	Average	Std
50	*Acc*	2.85E+00	5.19E-01	1.34E+00	1.75E-01	2.02E+01	3.97E+00	6.74E+02	1.91E+02
	NFE	4.88E+03	8.16E+00	4.88E+03	7.63E+00	4.88E+03	6.55E+00	4.88E+03	7.26E+00
100	*Acc*	2.46E-02	1.22E-02	1.99E-01	2.49E-01	7.53E-01	1.17E+00	8.90E+00	3.31E+01
	NFE	9.85E+03	1.09E+01	9.85E+03	8.76E+00	9.84E+03	9.65E+00	9.85E+03	9.55E+00
150	*Acc*	1.10E-05	6.56E-06	1.72E-01	2.76E-01	1.51E-07	3.11E-07	2.14E-06	7.41E-06
	NFE	1.98E+04	1.48E+01	1.98E+04	1.67E+01	1.98E+04	1.35E+01	1.98E+04	1.67E+01
200	*Acc*	4.09E-09	2.51E-09	1.60E-01	2.27E-01	1.71E-14	3.04E-14	8.79E-12	6.89E-13
	NFE	2.97E+04	1.79E+01	2.97E+04	3.00E+01	2.97E+04	2.16E+01	2.97E+04	2.39E+01
250	*Acc*	1.56E-12	7.88E-13	1.28E-01	2.69E-01	0	0	8.19E-12	9.25E-13
	NFE	3.97E+04	2.01E+01	3.97E+04	5.92E+01	3.97E+04	2.52E+01	3.97E+04	2.68E+01
300	*Acc*	6.44E-15	1.07E-15	1.30E-01	2.20E-01	0	0	*7.52E-12*	6.29E-13
	NFE	4.96E+04	3.42E+01	4.96E+04	1.04E+02	4.97E+04	23.55097	4.97E+04	2.65E+01

For the second scenario, when the number of cycles/iterations increases, the accuracy of the solutions improves accordingly. For a number of cycles greater than 100 the accuracy significantly decreases, which confirms the convergence of the DE algorithm and its ability to deal with multimodality.

Table **3** The third scenario. Results for the variable size of the search space.

SCENARIO III (no of variable variation, *library size =100, no of iterations=100)* ***Average ACCURACY and NFE for 30 runs***									
No. of variable	Metri c	F1 Ackley		F2 Griewank		F3 Rastrigin		F4 Schwefel	
		Average	Std	Average	Std	Average	Std	Average	Std
5	*Acc*	**2.66E-05***	*1.32E-04*	**4.99E-02**	*2.73E-02*	**4.60E-03 ****	*2.52E-02*	**6.40E-04**	*2.31E-03*
	NFE	**1.97E+04**	*3.71E+01*	**1.97E+04**	*3.08E+01*	**1.97E+04**	*2.03E+01*	**1.96E+04**	*6.96E+01*
10	*Acc*	**1.52E-06**	*8.32E-06*	**2.46E-01**	*3.00E-01*	**4.74E-16*****	*2.59E-15*	**2.36E-04**	*5.98E-04*
	NFE	**1.96E+04**	*2.46E+01*	**1.96E+04**	*2.46E+01*	**1.97E+04**	*2.43E+01*	**1.97E+04**	*2.85E+01*
20	*Acc*	**2.69E-04**	*1.32E-04*	**2.78E-01**	*3.05E-01*	**3.07E-06**	*2.66E-06*	**2.79E-04**	*5.14E-05*
	NFE	**1.97E+04**	*2.54E+01*	**1.97E+04**	*2.89E+01*	**1.97E+04**	*2.50E+01*	**1.97E+04**	*2.08E+01*
30	*Acc*	**1.41E-02**	*4.65E-03*	**2.93E-01**	*3.00E-01*	**4.24E-02**	*4.23E-02*	**1.59E-01**	*1.24E-01*
	NFE	**1.97E+04**	*2.10E+01*	**1.97E+04**	*2.94E+01*	**1.97E+04**	*2.79E+01*	**1.97E+04**	*2.40E+01*

** Modal value=0, No of 0s=19, Order of magnitude=-10; **Modal value=0, No of 0s=25, Order of magnitude=-10; *** Modal*

For the third test scenario, the dimension of the search space varies. Thus it is hard to give conclusive assessment from the table above, as the average accuracy of the solutions does not improve monotonically with the search spaces dimension, as is expected (higher dimension would correspond to larger error). Nevertheless, for the 5-variables test scenario, Ackley test function, the DE algorithm converges to the global minimum 19 times out of 30 runs. In this situation the mean of the accuracy doesnt reflect the good performance of the algorithm, therefore, the modal value (*the most frequent data value*) and the number of occurences of the modal value are presented too.

When we affirm that the algorithm finds the global optimum we refer to the situation where the accuracy of the final solution is computed as 0 as the variable values are of an order of magnitude smaller than -10 and the cosinus function returns 1. For those cases where the global minimum value is obtained we have given in the table the average order of magnitude for the variables values.

For 5 and 10 variables, Rastrigin test function, the DE algorithm finds the global minimum for more than 70% of the runs. An intriguing situation occurs for 10 variables, where the success ratio (29/30) is higher than for the 5 variables (2530). Also, as the local optima are less and more spaced apart for fewer variables, the premature convergence is more probable, and that could be an explanation of the situation previously described. The difference between the results for 5 and 10 variables cases also resides in the average number of cycles in which the global optimum is attained. So, the average number of cycles in which the optimum is attained is less than 40 cycles for 5 variables and less than 80 cycles for 10 variables. As the table shows, the average accuracy in the same number of evaluations (NFE) corresponding to 100 cycles and the better results obtained in 10 dimensions could be explained by the fact that the slower convergence for higher dimensions offers to the DE algorithm adequate time to overcome the local optima.

For a higher dimension of search space (e.g., 20 or 30 variables), the algorithm cannot provide the same accuracy in 100 cycles as it can for fewer variables. Therefore no occurrence of the global minimum value is recorded and in these situations the mean accuracy is conclusive. Excluding those situations when the zero values for the accuracy provoke a non-representative mean for the measurement of the central tendency, and therefore the evaluation of the algorithms performance could be misjudged, generally the expected behaviour is verified: the algorithm provides better results when the size of the search space is lesser.

The next experiments were conducted in order to compare the DE performance with the Particle swarm optimization algorithm (PSO) [16]. As in our previous experiments we chose the Rastrigin and Griewank test function, since DE provides interesting results for these functions. For each test the dimension of the search space varies as 5, 10, 20, and 30. PSO settings are 100 particles and 200 iterations, inertia weight linearly decreases from 0.9 to 0.4, learning coefficients = 2.0. For the space bounded by the range [min,max] the maximum velocity is 10% from max-min.

DE runs in 100 cycles with a maximum library of 100 genes and the elite is given by the best 20% from the current library. Each algorithm runs for 10 times and the accuracy is recorded. Independent-samples t-tests were conducted to compare PSO results and DE results There were significant differences in the accuracy metric for PSO and DE, considering 99% confidence intervals. The results summarized in Table 4 show that DE performs better for all the considered scenarios.

Table **4.** DE versus PSO. Comparisons' results for Ratrigin and Griewank test functions.

Griewank test function					**Rastrigin** test function				
No. of variable	**PSO**	DE	Tstat	*p*	No. of variable	PSO	DE	Tstat	*p*
5	1.32E-01	**4.81E-02**	3.72E+00	2.38E-03	**5**	1.40E+00	**0 ***	5.25E+00	2.64E-04
10	7.36E-01	**2.96E-01**	3.97E+00	1.62E-03	**10**	1.09E+01	**0* ***	9.23E+00	3.47E-06
20	1.08E+00	**2.82E-01**	6.32E+00	6.88E-05	**20**	2.77E+01	**3.50E-06**	1.58E+01	3.51E-08
30	1.40E+00	**3.10E-01**	1.21E+01	3.54E-07	**30**	6.25E+01	**3.04E-02**	2.30E+01	1.32E-09

*The most occurrence is given as for 90% in 40 cycles the 0 accuracy is found; * *90% in maximum 80 cycles the algorithm returns 0*

5. Conclusions

Among the many bio-inspired techniques which make up the fascinating landscape of Natural Computation, it is hard to find or to frame new paradigms that do not correspond closely to natural phenomena. Due to technological progress it is now possible to simulate, control and accelerate several natural processes in the laboratory. Among these semi-artificial protocols, we see Directed Evolution as a serious area of inspiration for computational techniques, due to its inner mechanisms and the structures it involves. The promise of developing a new branch in bio-inspired computing is substantiated by the richness of the techniques that such routines offer. Evolution and genetics represent the major sources of inspiration both in molecular engineering through Directed Evolution and in computer science through Evolutionary Algorithms. Comparing Directed Evolution and evolutionary algorithms, we observed that in terms of two common desiderata speed and the possibility of control the two instruments are similar. Our study emphasizes the novelty of the evolutionary paradigm inspired by Directed Evolution.

The DE algorithm is developed on the basis of the semi-artificial process of Directed Evolution of proteins. The strengths of the proposed technique are its ability to handle various optimization problems and the avenues it opens up towards a new research area. The proposed algorithm is not intended to compete with or surpass the other well-known evolutionary algorithms for optimization. Yet, even so, the preliminary results show that the DE technique is able to offer results that are at least as good as those given by a similar technique for the test problems we considered. For single-objective optimization, we analyzed the behavior of the DE algorithm for various test scenarios which involved different settings for library size, the number of cycles, and the dimension of the search space. The results suggest that the DE algorithm is consistent and viable as an optimization procedure. Compared to a similar bio-inspired technique, DE performs better than PSO for the test scenarios considered which involved popular test functions, each with a scalable dimension of the search space.

We note two major directions for further research: analysis of the DE algorithm's behaviour for multi-objective optimization, and in approaching real world problems.

References

[1] Cobb, R. E., Chao, R. and Zhao, H. (2013), *Directed evolution: Past, present, and future.* AIChE J., 59: 14321440. doi: 10.1002/aic.13995.
[2] Jckel, Christian, Peter Kast, and Donald Hilvert. *Protein design by directed evolution.* Annu. Rev. Biophys. 37 (2008): 153-173.
[3] Rubin-Pitel, S., et al. *Directed evolution tools in bioproduct and bioprocess development.* Bioprocessing for Value-Added Products from Renewable Resources: New Technologies and Applications (2006): 49-72.
[4] Moreno, Paula Cordero, Angel Goni Moreno, and Juan Castellanos Peuela. *Using directed evolution techniques to solve hard combinatorial problems.* In Proceedings of the Computer Science and Information Technologies Conference. CSIT 2009. Yerevan, Armenia. October 2009. 225-229.(2009).
[5] Berlik, Stefan. *Directed Evolutionary Algorithms by Means of the Skew-Normal Distribution.* S. Co. 2009. Sixth Conference. Complex Data Modeling and Computationally Intensive Statistical Methods for Estimation and Prediction. Maggioli Editore, 2009.
[6] Oates, M. J., D. W. Corne, and D. B. Kell. *The bimodal feature at large population sizes and high selection pressure: implications for directed evolution.* Recent Advances in Simulated Evolution and Learning (2003): 215-240.
[7] Voigt, Christopher A., et al. *Computationally focusing the directed evolution of proteins.* Journal of Cellular Biochemistry 84.S37 (2001): 58-63.
[8] Yokobayashi, Yohei, et al. *Directed evolution of trypsin inhibiting peptides using a genetic algorithm.* J. Chem. Soc., Perkin Trans. 1 20 (1996): 2435-2437.
[9] Weber, Lutz. *Applications of genetic algorithms in molecular diversity.* Current Opinion in Chemical Biology 2.3 (1998): 381-385.

[10] Arnold, Frances H. *Design by directed evolution.* Accounts of chemical research 31.3 (1998): 125-131.
[11] Cadwell, R. Craig, and Gerald F. Joyce. *Randomization of genes by PCR mutagenesis.* Genome research 2.1 (1992): 28-33.
[12] Stemmer, Willem PC. *Rapid evolution of a protein in vitro by DNA shuffling.* Nature 370.6488 (1994): 389-391.
[13] Gartner, Zev J. *Evolutionary approaches for the discovery of functional synthetic small molecules.* Pure and applied chemistry 78.1 (2006): 1-14.
[14] Biyani, Manish, et al. *Evolutionary Molecular Engineering to Efficiently Direct in vitro Protein Synthesis.* CELL-FREE PROTEIN SYNTHESIS (2012): 51.
[15] Park, Sheldon J., and Jennifer R. Cochran, eds. *Protein engineering and design.* Vol. 75. CRC press, 2009.
[16] Shi, Y., Eberhart, R. (1998, May). *A modified particle swarm optimizer. In Evolutionary Computation Proceedings*, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on (pp. 69-73). IEEE.
[17] De Castro, L.N. *Fundamentals of natural computing: basic concepts, algorithms, and applications.* CRC Press, 2006.
[18] Wilkins, M. R. et al., *From proteins to proteomes: large scale protein identification by two-dimensional electrophoresis and amino acid analysis.* BioTechnology14, 6165 (1996).
[19] Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y. P., Auger, A., Tiwari, S., *Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization.* KanGAL report, 2005005, (2005).

Corina Rotar
Department of Computer Science, Faculty of Science,
“1 Decembrie 1918” University of Alba-Iulia,
Romania
email: *crotar@uab.ro*