DE GRUYTER OPEN

JAISCR, 2017, Vol. 7, No. 3, pp. 183 – 200
10.1515/jaiscr-2017-0013

DIRECTED EVOLUTION – A NEW METAHEURISTC FOR OPTIMIZATION

Corina Rotar[1], Laszlo Barna Iantovics[2]

[1]*Department of Exact and Engineering Sciences, "1 Decembrie 1918" University*
Unirii street, no. 15-17, 510109 Alba Iulia, Romania
[2]*Department of Computer Science, "Petru Maior" University*
N. Iorga street, no. 1, 540088, Trgu Mure, ROMANIA

Submitted: 12th March 2016; accepted: 27th October 2016

Abstract

Recently, 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. Starting from the observation of natural processes at the macroscopic or microscopic level, various methods have been developed. 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 novel computational methods in the field of Natural Computing based on the semi-natural process, namely Directed Evolution. 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 computer scientists, this provide a strong source of inspiration in the search for efficient methods of optimization. The computational model that proposed 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. The paper describes a novel general algorithm, inspired by Directed Evolution, which is able to solve different optimization problems, such as single optimization, multiobjective optimization and combinatorial optimization problems.

Keywords: optimization, directed evolution, nature-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 the nature. A detailed account of the field is given in [25], 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. Through a diverse range of processes, phenomena, structures, and organizations, and thanks in no small measure to its complexity, nature constantly feeds research into developing new methods and paradigms that mimic it.

1.1 Motivation

Perspectives on nature are not limited to a single science: they are found in biology, physics, chemistry, sociology, etc. Yet, the computational methods that belong to computing inspired by nature lie beyond the boundaries of any individual branch of science. The computational researchers borrow various metaphors, theories and concepts, often selected from different areas, and linking them in order to design an efficient method for solving specific problems. Taking into account the proposed goal, the designer (the computer scientist) affords herself the freedom to abstract, simplify and re-interpret the original sources of inspiration, so that a perfect mapping between structures, natural phenomena and the product (the paradigm, method or technique) in fact proves difficult to achieve. An example of this is the rich complex of genetic algorithms inspired by the theory of neo-Darwinian natural selection theory, which encapsulates Mendelian genetics (heredity theory). As defined in [26] the artificial model transcends the borders of evolutionary biology, through making use of various mechanisms that allow an acceleration of the evolutionary process: these mechanisms are necessary in order to solve the problem in question, but thereby render the model different from the original source. In addition, the spontaneity of nature, whether established or as yet unexplained, cannot be fully mimicked through exact formulas. A genetic algorithm is equipped with additional control parameters or mechanisms to achieve convergence towards the desired solution. In a natural evolutionary process, such strict control is absent. That the computational entities evolve under strict control, contrary to the natural evolution of biological entities, is a discrepancy between natural phenomena and artificial processes which seemingly cannot be surpassed.

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. Whereas nature evolves spontaneously, but over a very long period of time, the lab allows for controlled evolution of biological entities over a short time. Thus 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 [25], 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 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.

1.2 Background

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. A first attempt to propose a novel technique inspired by Directed Evolution protocol, dedicated to solving several single objective optimization problems, is presented in [6]. The emergent nature of Directed Evolution is anticipated in [7], *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 [8]. Moreover, the evolutionary computation techniques are applied for optimizing the process of Directed Evolution or else they are exploited as *helpful tools* [9], [10].

2 Directed Evolution versus Natural Evolution

Natural evolution refers to the complex phenomena by which a species adapts in an almost intelligent manner to environmental conditions. Evolution is possible due to variation and it is guided by natural selection. These propositions were formulated by Charles Darwin in his famous work *On the Origin of Species* [17].

The evolutionary principles gradually became accepted among the academic community, and later the Darwinian ideas spread and became part of the collective mentality; nowadays evolutionary theory belongs to the formal education given through biology textbooks. Starting with the above-mentioned work, the roots of evolutionary theory have expanded and been supplemented by new ideas supported by scientific demonstrations in the field of biology.

If technical progress in the 19th century did not allow a more detailed analysis of the underlying structures of natural evolution, the twentieth century brought a remarkable change in this direction. Due to later discoveries, the original Darwinian Theory required revisions and additions, so that modern evolutionary theories or neo-Darwinian theory gradually came to integrate the principle of genetics. In his *The Genetical Theory of Natural Selection* [18], Fisher introduced the term *population genetics*, thereby proposing a new branch of biology which combines Mendelian genetics with Darwinian natural selection.

In the same period, in his *Evolution: The Modern Synthesis* (1942) [19], Huxley lay the foundation for the modern evolutionary paradigm, introducing the term *modern evolutionary synthesis* to refer to the unification of Darwinian principles and genetics. The discovery of the structure of DNA (heredity molecules) by Watson and Crick (1953) is one of the most remarkable discoveries of the twentieth century and was a turning point in the development of molecular biology. An overview of the dynamics of biology, from Darwin to Crick, reveals a major shift in perspective, from the macroscopic level to the most sophisticated level of cell biology. Nowadays, the latest discoveries in the fields of biology and chemistry and the rapid progress of technology have made possible the simulation of evolution in the laboratory.

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).

The ‘central dogma’ has foregrounded the simple formula of gene flow from DNA to protein. Protein synthesis is one of the most fundamental biological processes by which individual cells construct their specific proteins, and proves to be an area of interest in molecular engineering, so that Directed Evolution is obviously *used* at the protein level.

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 for subsequent iterations of diversification and selection. The process is repeated until the desired improvement is achieved. This powerful engineering tool is an iterative strat-

egy which includes at each round three major steps: amplification, diversification and selection [2-3]. A survey of Directed Evolution is given in [1].

Even if Directed Evolution mimics natural evolution, there are several features that differentiate the two processes. Firstly, Directed Evolution represents a process which is inspired by natural evolution and is recreated in an artificial environment. From this point of view, the necessary time for obtaining the expected results in laboratory is considerably shorter, the complete procedure taking just few days contrasting to the epochs over which the natural process occurs. Secondly, absent external interference, the natural process happens spontaneously by transmitting genetic traits from one generation to another, and mutations and selection of the adapted individuals. In the case of Directed Evolution, as the name implies, the entire process is controlled in the laboratory by various mechanisms of diversification, amplification and selection schemes that are designed according to the engineer's goal. Therefore, irrespective of its real purpose, Directed Evolution is in essence superior to the natural process from the perspective of time and the possibility of controlling the output.

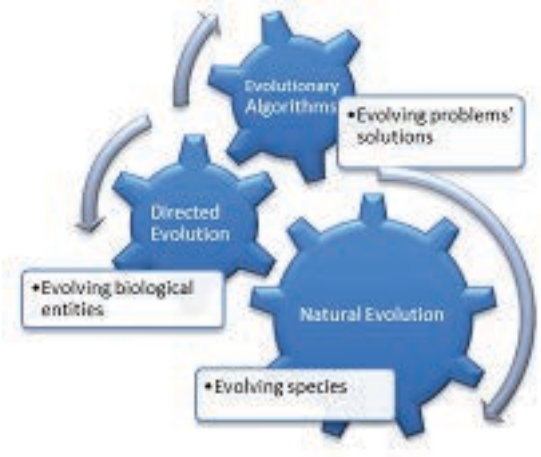

Figure 1. Synthetic description of the relations between natural and artificial paradigms

Natural evolution and genetics are major sources of inspiration in two different branches of science: Molecular Engineering through Directed Evolution, and Computer Science through Evolutionary Algorithms. By correlating directed evolution and evolutionary algorithms, it is noticeable that at least in terms of two aspects, time and control, the two instruments are very close. Just as Directed Evolution surpasses natural evolution in respect of *time* and *control*, so Evolutionary Algorithms achieve the desired results in a short time, by directing the evolution through sophisticated and ingenious mechanisms. A common desideratum of evolutionary techniques and Directed Evolution procedures is to get the best outcomes in a short time by following the principles of natural evolution. Based on these considerations, a computational model inspired by Directed Evolution can be proposed as a viable paradigm in *Natural Computing*.

3 The Metaphor of Directed Evolution

The Directed Evolution [11] 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 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 methods of random mutagenesis is error-prone polymerase chain reaction (PCR) [12] 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 [13], 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 [16]. The proteins that are expressed by the produced library of genes require linkage to the correspondent genetic code [15], 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 iden-

tify 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 simultaneously 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 [14]. The entire procedure repeats until the goal is achieved – for example, obtaining a protein with specific functionality.

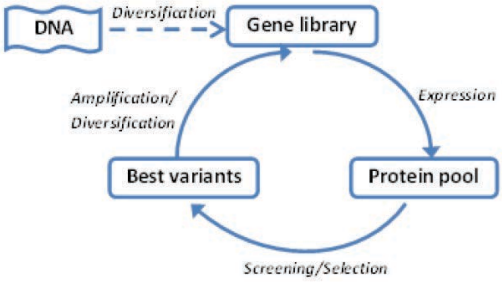

Figure 2. Schematic overview of directed evolution cycle

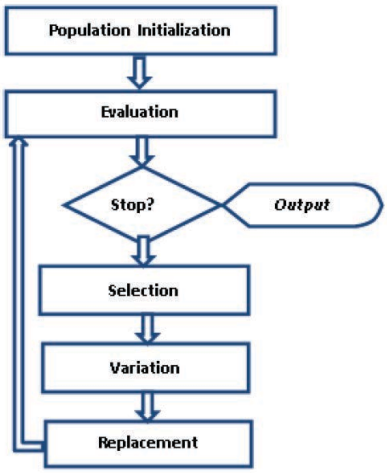

Figure 3. Schematic overview of an evolutionary algorithm

A brief description of Directed Evolution reveals several features that are remarkably similar to the process described by *evolutionary algorithms*. Defined in the 1950s as methods of solving search and optimization problems, these biological-inspired computational techniques have been rapidly developed according to the principles of genetics and the evolutionary metaphor. Evolutionary computation comprises four main areas: Genetic Algorithms, Evolutionary Strategies, Evolutionary Programming, and Genetic Programming, to which have recently been added other biologically inspired techniques (Particle Swarm Optimization [23], Ant Colony Optimization [27], Artificial Immune Systems [28], and so on).

Although the literature is extensive, the research focuses on developing new performant algorithms or finding new problems where the evolutionary techniques prove to be efficient. In this landscape, but also from the perspective of similarities with the procedures of Directed Evolution, a new Directed Evolution–inspired paradigm seems to be appropriate.

Next, a scheme of Directed Evolution for problem solving is described. The system contains:

a 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)

b Computing procedures which describe the corresponding processes from Directed Evolution (Diversification, Expression, Screening/Selection)

The main elements (structures and modules) of the model described in Figure 4 are detailed next:

Table 1. Description of the involved structures in natural and artificial paradigma

Structures		Desciption
Natural Paradigm	Genes (DNA)	- sequence of nucleotides in DNA, which is the functional unit of inheritance; code for making proteins: the information that is stored in DNA is partially transferred into proteins using RNA as an intermediary(transcription, translation)
Natural Paradigm	Proteins	-large biological molecules consisting of chains of amino acids; these are generated through the process of gene expression in two stages: **transcription** and **translation**
Artificial Paradigm	Genes (DNA)	- possible solutions from the search space, **numerically codified**
Artificial Paradigm	Proteins	- solutions in the objective space (gene's product) - *the protein synthesis procedure is comparable to the process of mapping the search space to the objective space*

Modules:

a **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.

b **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.

c **Screening/Selection** mimics the corresponding screening and selection process from the natural paradigm, aiming to identify those elements which respond to the problem's objectives by establishing correspondence between the genes and the expressed proteins.

d **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 by mutagenesis.

The aim of the evolutionary method here developed is to evolve the genetic code that corresponds to the beneficial proteins that conform to the problem's objectives. The whole process simulates the procedures and the controlling mechanism from the paradigm of Directed Evolution. The first phase consists in generating a diverse library of genes (possible solutions in the search space) which are expressed for obtaining the proteins' library (*fitness landscape*). The selection procedure identifies the most promising structures (the proteins) according to the given problems' objectives. The genes that correspond to those proteins are further subject to amplification and diversification, and thus a new library of genes is constructed. The cycle repeats until the outcome of the entire process is satisfactory.

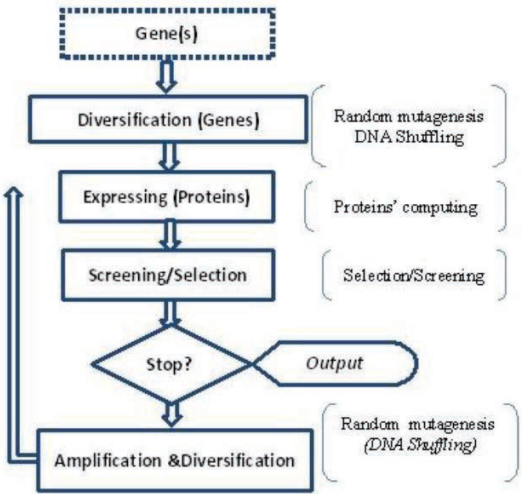

Figure 4. Schematic overview of a computational directed evolution algorithm

4 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.

4.1 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: $(x_1, x_1^1, x_2, x_1^2, ..., x_n)$ (Figure 5).

The natural protein represents a chain of aminoacids. The corresponding objectives' values computed on the basis of the genes' codification form the protein structures $(f_1 x_1^1, f_2, x_1^2, ..., f_m)$, which correspond to the *m-dimensional* vector as in Figure 6.

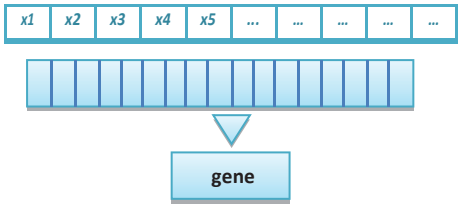

Figure 5. Gene structure

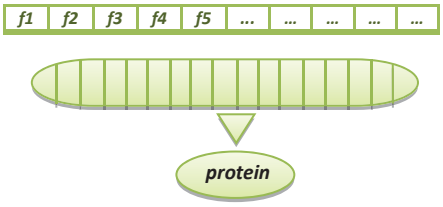

Figure 6. Protein structure

4.2 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]) (fig. 7). 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 (fig. 8).

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

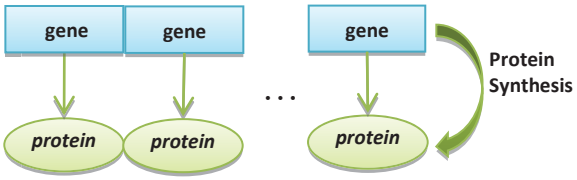

Figure 7. Gene expression. Protein synthesis. (For optimization problems without constrains)

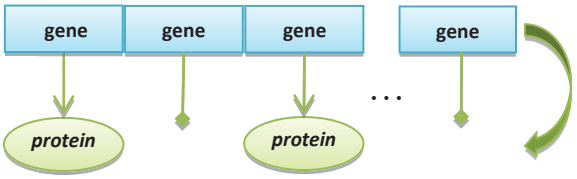

Figure 8. Gene expression. Protein synthesis. (For optimization problems involving constrains)

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, obtained using Translate function *P=Translate(G):*

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

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

Function **Translate** (G_i, Protein$_i$)

for each objective j=1 to m

Protein$_i(j) \leftarrow f_j(G_i)$ //compute the jth objectives

end for End

Procedure **Expressing for** each gene $G_i \in DNA$

Protein$_i \leftarrow$ Translate(G_i) //protein expressing

end for End

4.3 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 screening for survival of the improved sequences can be performed within the variation procedures after each new gene is generated from the original genes. Therefore, in DNA Shuffling and Mutagenesis, the new genetic sequences may be inspected and compared with the original ones and the outcome of the direct comparisons may be further included into the Gene Library. Nevertheless, this above-described procedure, even though it is straightforward, would significally increase the selection pressure, and should be generally avoided. This screening process may be required for some optimization problems where the risk of the premature convergence is limited. Also, because this phenomenon cannot be checked beforehand, the selection mechanism that is described next would be more suitable for a general optimization technique.

The selection mechanism will favour the best candidates from the genetic library. Depending on the optimization problem, whether single or multiobjective, the selection procedure gathers either the elite among the genetic pool according to the single objective values, or the sub-set of the Pareto nondominated solutions. In both cases 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. As for multiobjective optimization problems, the number of the nondominated candidates gives the size of the selected elite, where for a single-objective optimization this size should be defined a priori. 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. To put it differently, the screening process analyzes each newly created gene and determines if it replaces the original one.

Procedure **Selection**

Input DNA_new

DNA$\leftarrow$ *Elite*(DNA_New)

Output DNA

End

4.4 Diversification: Mutagenesis and DNA Shuffling

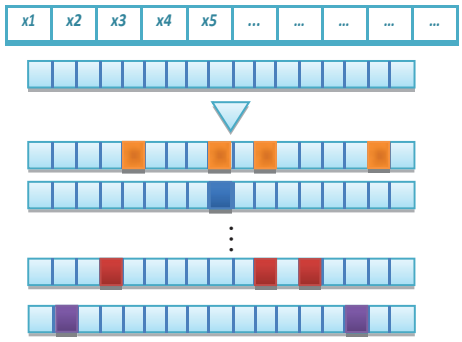

Figure 9. Mutagenesis. Mutated DNA sequence may vary differently, according to the mutation frequency

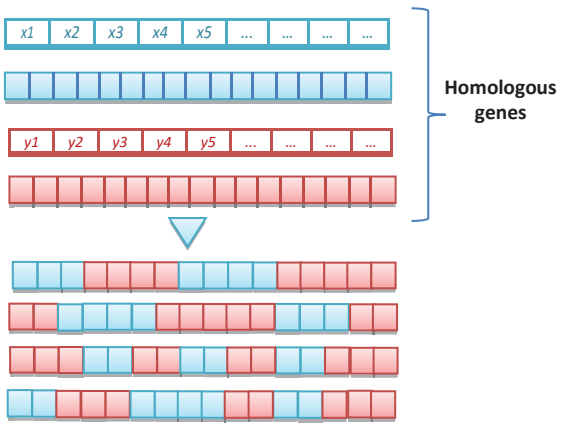

Figure 10. Shuffling of homologous genes

Considering DNA=$\{G_1,\ldots,G_S\}$ - the genes' library and Π=$\{Protein_1,\ldots,Protein_S\}$ – the 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.

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

$$frequency = S_{\max}/S,$$

where S_{max} is the maximum size of the library and S represents the current size of the library.

This value measures to what extent each variant proliferates into the genetic pool. Depending on the type of the optimization problem, whether single or multiobjective, the frequency is a constant value during the evolutionary process or else varies according to the variable size of the current population.

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.

Procedure **Mutagenesis**

Input DNA – *current genetic library*

for each gene $G_i \in$DNA

for k=1 to frequency

M $\leftarrow$ Mutate (G_i)

Copy(G_New$_i$, M)

end for

end for

Output DNA_new //*new genetic library*

End Mutagenesis

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 **DNA_Shuffling**

Input DNA – *current genetic library*

//copy first S genes into the new DNA

for k=1 to S

Copy(G_new$_k$,G$_k$)

end for

for k=S+1 to Smax

//binary tournament selection

BinarySelection G$_i \in$DNA

//Search homologous gene Hi$\in$DNA

H$_i \leftarrow$*Homologous* (G$_i$)

New$\leftarrow$*GeneticShuffle* (G$_i$,H$_i$)

Copy(G_new$_i$,*New*)

end for

Output DNA_new // *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 or to the attaining of the desired quality of the protein*

5 Experimental results - algorithm's behaviour

In order to evaluate the behaviour of the DE algorithm, we considered several popular test functions: *Ackley, Griewank, Rastrigin* and *Schwefel*, which are scalable as regards the number of variables. The test functions are described in [30].

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 fitness evaluations is also recorded.

We considered three test scenarios to observe the algorithm's performance when the size of library, the number of cycles and the number of variables varies:

a **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.

b **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%), the number of variables is 30.

c **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.

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.

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.

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 space's 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 doesn't 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.

We consider that the algorithm finds the global optimum when the accuracy of the final solution is computed as 0 as the variable values are of an order of magnitude higher 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 Rastrigin test function with 5 and 10 variables, the DE algorithm finds the global minimum for more than 70% of the runs. An interesting situation occurs when 10 variables were considered, where the success ratio (29/30) is higher than for the 5 variables' case (25/30). 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

Table 2. SCENARIO I: *Average ACCURACY and NFE for 30 runs* (library size varies, constant *no. of iterations=100, constant no. of variables=30)*

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

Table 3. SCENARIO II: *Average ACCURACY and NFE for 30 runs* (no of iteration variation, *library size =50, constant no. of variables=30)*

Iteration No.	Metric	F1 Ackley		F2 Griewank		F3 Rastrigin		F4 Schwefel	
		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

Table 4. **SCENARIO III:** ***Average ACCURACY and NFE for 30 runs*** **(no of variable variation,** ***library size =100, no of iterations=100)***

Iteration No.	Metric	F1 Ackley		F2 Griewank		F3 Rastrigin		F4 Schwefel	
		Average	Std	Average	Std	Average	Std &modala	Average	Std
5	*Acc*	**2.66E-05** *(a)*	*1.32E-04*	**4.99E-02**	*2.73E-02*	**4.60E-03** *(b)*	*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** *(c)*	*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*

(a)- Modal value=0, No of 0s=19, order of magnitude=-10
(b)- Modal value=0, No of 0s=25, Order of magnitude=-10
(c)- Modal value=0, No of 0s=29, Order of magnitude=-10

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 algorithm's performance could be misjudged, generally the expected behaviour is verified: the algorithm provides better results when the size of the search space is lesser.

6 Experimental results - algorithm's performance

The following paragraph is organized in three parts, corresponding to three types of optimization problems we took into consideration: single objective optimization, multiobjective optimization and combinatorial optimization. Each set of experiments is detailed and the proposed algorithm's performance is compared with a state of art technique for the considered problem. Therefore, for single objective optimization, we choose a very popular bio-inspired technique, namely Particle Swarm Optimization [23], which previously proved its performance in solving these type of problems. For the challenging multiobjective optimization problem, we conducted various experiments, considering popular test problems and comparing the proposed algorithm's performance with a state of art algorithm, namely Non-dominated Sorting Genetic Algorithm 2 (NSGA2) [22]. Finally, for combinatorial optimization problem (1/0-*Knapsack Problem)*, the DE algorithm's results are compared with the results obtained through dynamic programming technique, considering various scenarios which involves several cases of test data.

6.1 Single objective optimization

The next experiments were conducted in order to compare the DE performance with the *Particle swarm optimization algorithm* (PSO) [23]. PSO represents one of the best competitors in the landscape of nature-inspired computing techniques for single objective optimization. Also, both PSO and DE are population-based optimization technique, and consequently, it allows a fair comparison.

As in our previous experiments, where the algorithm's consistency was studied, we chose the Rastrigin and Griewank test function, since both functions are highly multimodal and challenge the algorithms to not be trapped in one of the local optima. Also, 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 5 show that DE performs better than PSO for all the considered scenarios.

Table 5. DE versus PSO. Comparisons' results for Ratrigin and Griewank test functions

Griewank test function				
No. of var.	**AVERAGE ACCURACY**		Tstat	p
	PSO	**DE**		
5	1.3e-01	**4.8e-02**	3.7e+00	2.4e-03
10	7.4e-01	**2.9e-01**	3.9e+00	1.6e-03
20	1.1e+00	**2.8e-01**	6.3e+00	6.9e-05
30	1.4e+00	**3.1e-01**	1.2e+01	3.5e-07
Rastrigin test function				
No. of var.	**AVERAGE ACCURACY**		Tstat	p
	PSO	**DE**		
5	1.4e+00	**0 ***	5.2e+00	2.6e-04
10	1.1e+01	**0* ***	9.2e+00	3.5e-06
20	2.8e+01	**3.5e-06**	1.6e+01	3.5e-08
30	6.2e+01	**3.0e-02**	2.3e+01	1.3e-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***

6.2 Multiobjective optimization

A more challenging task for an optimization technique is to properly solve multiobjective optimization problems. Therefore, in order to investigate the performance of DE in a multiobjective context, we used several well-known test problems: ZDT1, ZDT2, ZDT3 [20] as the two objective test functions with 30 variables, and the DTLZ1, DTLZ2 and DTLZ3 problems [21] with 2, 3 and 4 objectives and the corresponding number of variables.

Table 6. Description of the test scenarios for multiobjective optimization

Scenario No.	Test function	No. of obj.	No. of var.
1	**ZDT1**	2	30
2	**ZDT2**	2	30
3	**ZDT3**	2	30
4	**DTLZ1 2**	2	6
5	**DTLZ1 3**	3	7
6	**DTLZ1 4**	4	8
7	**DTLZ2 2**	2	11
8	**DTLZ2 3**	3	12
9	**DTLZ2 4**	4	13
10	**DTLZ3 2**	2	11
11	**DTLZ3 3**	3	12
12	**DTLZ3 4**	4	13

Table 6 describes 12 test scenario where the number of objectives and the number of variables varies accordingly for the popular test problems we considered.

For performance assessment, we compute the hyper-volume metric (HV) [21]. The hyper-volume metric corresponds to the size of the objective space which contains the solutions which are Pareto-dominated by at least one of the members of the set, and it represents a widely-used metric to evaluate the performance of a multiobjective optimization technique. The smaller the hypervolume value is, the better outcomes the algorithm provides.

For an objective comparison with the popular algorithm NSGA2 [22] we set the following parameters for the DE: maximum size of the Gene Library is set to 50, initial size of the library is 10 (20% from the maximum size) and the number of cycles per each run is set to 100. NSGA2's parameters are 100 individuals and 100 generations per run. These settings assure the same maximum number of function evaluations for both algorithms. As the elite size depends on the number of Pareto nondominated solutions, it cannot be exactly equal to 20% of the maximum size of the library. Therefore, for the DE algorithm, the number of function evaluations is lesser to some extent than for NSGA2.

The algorithms runs for 20 times and the hypervolume values are computed.

The results from Table 7 reveal several conclusions. For the first test function ZDT1 – convex Pareto front, NSGA2 performs better than the DE algorithm. For the following bi-objective optimization test problem (*concave Pareto front*) the DE defeats its competitor and for ZDT3 (*discontinued Pareto front)* DE and NSGA2 have similar performance.

Table 8. Statistical results. Synthesis *(+ better, - weaker, = similar)*

Test function	Objectives /variables	DE algorithm	NSGA2
ZDT1	2/30	-	+
ZDT2	2/30	+	**-**
ZDT3	2/30	=	**=**
DTLZ1	2/6	-	+
	3/7	-	+
	4/8	+	**-**
DTLZ2	2/11	-	+
	3/12	-	+
	4/13	-	+
DTLZ3	2/11	+	**-**
	3/12	+	**-**
	4/13	+	**-**

For DTLZ1 with 2, 3 and 4 objectives, DE loses the competition in the first two scenarios but for 4 objectives it performs better. For a conclusive report we considered an extra scenario for the DTLZ1 test function (*many local Pareto fronts*), in order to observe the behaviour of the DE algorithm for the bi-objective case but in a different setting. Therefore, we set the maximum number of function evaluations to 20,000: NSGA2 runs with a population of 100 individuals for 200 generations and DE runs with a maximum library of 50 genes in 200 cycles. The results confirm that DE has a disadvantage

Table 7. Hypervolume metric: Average and standard deviation values. Statistical tests for the considered scenarios

No.	DE	NSGA2	*Tstat*	*p*
1	7.48e-01(*2.45e-02*)	**7.76e-01**(*2.00e-02*)	-4.05e+00	1.25e-04
2	**6.24e-01**(*3.21e-02*)	5.21e-01(*7.69e-02*)	*5.67e+00*	*3.31e-06*
3	**5.90e-01**(*1.63e-02*)	5.80e-01(*2.40e-02*)	1.57e+00	6.26e-02
4	6.10e-01(*8.09e-02*)	**6.70e-01**(*1.10e-01*)	-2.01e+00	2.62e-02
5	5.87e-01(*5.93e-02*)	**7.05e-01**(*1.64e-01*)	-3.11e+00	2.38e-03
6	**5.64e-01**(*9.20e-02*)	1.33e-01(*7.57e-02*)	1.66e+01	4.89e-19
7	7.50e-01(*1.80e-02*)	**7.72e-01**(*9.99e-05*)	-5.55e+00	1.18e-05
8	7.96e-01(*5.27e-02*)	**8.91e-01**(*3.73e-03*)	-8.22e+00	5.62e-08
9	7.74e-01(*3.82e-02*)	**9.24e-01**(*8.01e-03*)	-1.76e+01	2.34e-14
10	**1.94e-01**(*5.79e-02*)	1.05e-01(*4.11e-02*)	5.76e+00	8.86e-07
11	**1.71e-01**(*6.38e-02*)	1.04e-01(*4.08e-02*)	4.05e+00	1.54e-04
12	**4.79e-02**(*2.28e-02*)	4.36e-03(*1.11e-03*)	8.77e+00	2.09e-08

when the number of cycles is smaller as compared to its competitor. But, again, when the number of cycles increases, the DE algorithm overcomes its shortcoming. The cause of this behavior could be the nature of the DTLZ1 problem, which is highly multimodal. Consequently, for an insufficient number of cycles the DE may not have the strength to converge to the global optimum Pareto front.

For DTLZ2 the results clearly show that NSGA2 performs better than DE independently of the number of objectives. However, for DTLZ3, which has the same real Pareto front as the former, but is more challenging than DTLZ2 since it introduces multiple local Pareto optimal fronts, DE outperforms NSGA2.

Table 8 presents the synthesis of the statistical tests.

6.3 1-0 Knapsack problem

The 1-0 Knapsack problem is a combinatorial problem defined as follows:

Given n items of different weights w_i, $i = 1,...,n$ and capacities c_i, $i = 1,...,n$ and a knapsack of maximum capacity *Capacity*, find the optimum solution consisting in the set of items with maximum sum of weights which do not surpass the maximum capacity of the knapsack.

The Knapsack problem is of particular interest as it is very often met in real world tasks and is a combinatorial optimization problem.

In our approach, each gene represents a binary sequence which reflects the selection or not of a specific object

G=(x_1, x_2, ..., x_n),

where $x_i = \begin{cases} 1 & \text{if } i^{th} \text{ item is selected;} \\ 0 & \text{otherwise.} \end{cases}$

The genetic shuffling does not strictly depend on the nature of the gene, as the mechanism involves copying the genes from the original genes. Nevertheless, the mutation is altered to reflect the binary codification and consists in switching the genes' values from 1 to 0 and vice versa.

The Knapsack problem is formulated as a single-objective optimization where the function to be minimized is computed as follows

$$f_{knapsack}(G) = \frac{\sum_{i=1}^{n} w_i - \sum_{i=1}^{n} G.x(i) \cdot w_i}{\sum_{i=1}^{n} w_i} +$$

$$+\alpha \cdot \frac{Abs(Capacity - \sum_{i=1}^{n} G.x(i) \cdot c_i)}{Capacity}.$$

The protein represents the real value obtained by computing the function $f_{knapsack}$ of the corresponding gene. Furthermore, due to the binary representation of the genetic material, the similarity of two genes is measured through the Hamming distance.

Experiments

Several cases of test data sets were taken into account [24]:

1 *uncorrelated,*

2 *weakly correlated,*

3 *strong correlated,*

4 *inverse strong correlated,*

5 *almost strongly correlated,*

6 *subset sum.*

The items' weights and capacities are generated in $[1 \ldots 100]$. The maximum capacity of the knapsack is set to 500. For each data set the algorithm runs for 30 times, for 50 cycles, with a maximum library of 100 genes of which 20% forms the elite.

At each run we recorded the weight of the best solution. For comparing the solutions found by the DE algorithm we use as the exact solution those that are found by using the conventional *Dynamic programming* technique.

Table 9. Knapsack problem. Results

Data Set Type	Average max. weight	Solution (max. weight)	Ratio	Insucess /Success	Success Ratio
(1)*	1905	1905	**1**	0/30	1
(2)*	**642.4**	**644**	**0.997**	**9/21**	**0.7**
(3)*	799.97	800	**0.99**	1/29	**0.97**
(4)*	448.53	450	0.997	3/27	**0.9**
(5)*	**790**	**800**	**0.987**	**30/0**	**0**
(6)*	500	500	**1**	0/30	**1**

**(1)-Uncorrelated, (2)-Weakly Correlated, (3)-Strongly Correlated, (4)-Inverse Strongly Correlated, (5)-Almost Strongly Correlated, (6)-Subset Sum.*

The results shown in Table 9 prove that the DE is able to deal with the NP-complex problem as a Knapsack problem.

7 Conclusions and discussions

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 highlights the novelty of the evolutionary paradigm inspired by Directed Evolution and extends the proposed DE technique for solving more than one type of optimization problems.

The DE algorithm was 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 the similar techniques for the test problems we considered. PSO is proven effective in solving optimization problems with a single objective while NSGA2 is recognized as one of the most popular algorithms for multiobjective optimization. In order to evaluate the performance of the proposed algorithm, we chose several optimization problems with one criterion, popular test problems for multiobjective optimization, and the generic 1-0 Knapsack problem. To provide an insight for evaluation, for each category of problem we provided the results obtained in similar settings with state-of-the-art algorithms. For the Knapsack problem the solutions found by the DE algorithm are compared with the global optima given by a conventional algorithm.

Firstly, for single-objective optimization, we reported 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.

Secondly, for multi-objective optimization, the DE algorithm overcomes NSGA2 for half of the considered scenarios. Even though NSGA2 overcomes DE for the DTLZ2 problem, it is notable that DE offers better results for a more difficult test problem like DTLZ3. Also, while NSGA2 offers better results for DTLZ1 problem with 2 and 3 objectives, and when 4 objectives are specified, DE defeats its competitor.

Thirdly, for the popular 1-0 Knapsack problem we investigated the DE algorithm for different random test data and we compared the results with the exact solutions found by using a conventional technique. With minimum intervention into the DE algorithm, which involves the binary codification of the DNA sequences, similarity measurement of the genes by using Hamming distances, and an evaluation function which reflects the problem's objective, DE can be applied with success. Except for the *almost strongly correlated test data*, when DE is not able to find the global optimum in any of the 30 runs, and for the weakly *correlated test data* when DE's success ratio is 0.7, for the other scenarios DE provides a success ratio greater than 0.9. For the most general test case when uncorrelated data are involved, DE finds the global optimum at each run in less than 50 cycles.

As further research, we note two major directions: analysis of the DE algorithm's behaviour for optimization in a dynamic environment, and in approaching real world problems.

References

[1] Cobb, R. E., Chao, R. and Zhao, H., Directed evolution: Past, present, and future. AIChE Journal, 59, 2013, p. 1432–1440.

[2] Jckel, C., Kast P., and Hilvert D., Protein design by directed evolution, Annu. Rev. Biophys, 37, 2008, p. 153-173.

[3] Rubin-Pitel S., et al., Directed evolution tools in bio-product and bioprocess development, In Bioprocessing for Value-Added Products from Renewable Resources: New Technologies and Applications, 2006, p. 49-72.

[4] Moreno, P. C., Moreno A. G., and Peuela C. J., Using directed evolution techniques to solve hard combinatorial problems, Proceedings of the Computer Science & Information Technologies Conference. CSIT 2009, p. 225-229.

[5] Berlik, S., Directed Evolutionary Algorithms by Means of the Skew-Normal Distribution, In S. Co. 2009 Sixth Conference. Complex Data Modeling and Computationally Intensive Statistical Methods for Estimation and Prediction. Maggioli Editore, 2009, p.67.

[6] Rotar, C., Directed Evolution-a Bio-inspired Optimization Technique, Proceedings of International Conference on Theory and Applications in Mathematics and Informatics, Alba Iulia, 2015.

[7] 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, p. 215-240.

[8] Voigt C. A., et al., Computationally focusing the directed evolution of proteins, Journal of Cellular Biochemistry, 2001, p. 58-63.

[9] Yokobayashi, Yohei, et al., .Directed evolution of trypsin inhibiting peptides using a genetic algorithm, J. Chem. Soc., Perkin Trans. 1.20, 1996, p. 2435-2437.

[10] Weber L., Applications of genetic algorithms in molecular diversity, Current Opinion in Chemical Biology 2.3, 1998, p. 381-385.

[11] Arnold F. H., Design by directed evolution, Accounts of chemical research 31.3, 1998, p. 125-131.

[12] Cadwell R. C., and Gerald F. J., Randomization of genes by PCR mutagenesis, Genome research 2.1, 1992, p. 28-33.

[13] Stemmer W. PC., Rapid evolution of a protein in vitro by DNA shuffling, Nature 370.6488, 1994, p. 389-391.

[14] Gartner Z. J., Evolutionary approaches for the discovery of functional synthetic small molecules, Pure and applied chemistry 78.1 2006, p. 1-14.

[15] Biyani M., et al., Evolutionary Molecular Engineering to Efficiently Direct in vitro Protein Synthesis, CELL-FREE PROTEIN SYNTHESIS, 2012, p. 51.

[16] Park S. J., and Cochran J. R., eds. Protein engineering and design. Vol. 75. CRC press, 2009.

[17] Darwin Ch., and Beer G., The origin of species. Oxford: Oxford University Press, 1951.

[18] Fisher R. A., The genetical theory of natural selection. , 1958, available online at https://archive.org

[19] Huxley J., Evolution. The Modern Synthesis, 1942. available online at www.ehudlamm.com/huxley.pdf

[20] Zitzler E., et al., Performance assessment of multiobjective optimizers: An analysis and review. Evolutionary Computation, IEEE Transactions on, 7(2), 2003, p. 117-132.

[21] Zitzler E., Deb, K., Thiele, L., Comparison of Multiobjective Evolutionary Algorithms: Empirical Results, Evolutionary Computation, vol. 8 no, 2, 2000, p. 173-195.

[22] Deb K., et al., A fast and elitist multi-objective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation, 6 (2), 2002, pp. 182-197.

[23] Shi, Y. and Eberhart, R., A modified particle swarm optimizer, In Evolutionary Computation Proceedings, IEEE World Congress on Computational Intelligence., 1998, pp. 69-73.

[24] Pisinger D., Where are the hard knapsack problems?, Computers & Operations Research 32.9, 2005, p. 2271-2284.

[25] De Castro, L.N., Fundamentals of natural computing: basic concepts, algorithms, and applications. CRC Press, 2006.

[26] Mitchell, M. An introduction to genetic algorithms. MIT press, 1998.

[27] Dorigo M., Birattari M., and Stutzle T., Ant colony optimization, Computational Intelligence Magazine, IEEE 1.4, 2006, p. 28-39.

[28] De Castro L.N., and Timmis J., Artificial immune systems: a new computational intelligence approach, Springer Science & Business Media, 2002.

[29] Wilkins M. R. et al., From proteins to proteomes: large scale protein identification by two-dimensional electrophoresis and amino acid analysis, BioTechnology 14, 1996, p. 61–65.

[30] Adorio E. P., Diliman U., MVF-Multivariate Test Functions Library in C for Unconstrained Global Optimization, 2005, available online at http://www.geocities.ws/eadorio.

Corina Rotar is associate Professor at Sciences and Engineering Faculty from "1 Decembrie 1918" University of Alba Iulia. Her research activity focuses on bio-inspired computing, computational intelligence, and multi-objective optimization. In 15 years of activity, she performs teaching activity on several subjects as artificial intelligence, evolutionary computation, object-oriented programming, coordinates the scientific events, participates in research projects and directs the administrative activity of the Exact Science and Engineering Faculty. She is the author of more than 50 research articles and 2 books. She holds a Ph.D. from "Babes-Bolyai" University in Computer Science.

Associate Professor **L.B. Iantovics** received Ph.D. in Artificial Intelligence at Babes-Bolyai Univ. of Cluj-Napoca and finished a Postdoctoral study in Artificial Intelligence at Alexandru Ioan Cuza University of Iasi. Actually he teaches at Petru Maior University and University of Medicine and Pharmacy from Tg. Mures. His main research interests includes the intelligent systems and computational intelligence, topics on that he published dozens of papers and contributed to research projects as project director or researcher. He was chair and scientific organizer of International Conferences. He is director of the "Advanced Computational Technologies" from Petru Maior University.