

Do not Choose Representation just Change: An Experimental Study in States based EA

Maroun Bercachi, Manuel Clergue, Philippe Collard, Sébastien Verel

▶ **To cite this version:**

Maroun Bercachi, Manuel Clergue, Philippe Collard, Sébastien Verel. Do not Choose Representation just Change: An Experimental Study in States based EA. GECCO '09 the 11th Annual conference on Genetic and evolutionary computation, Jul 2009, Montréal, Canada. pp.1799-1800, 10.1145/1569901.1570168 . hal-00945549

HAL Id: hal-00945549
https://hal.science/hal-00945549

Submitted on 17 Feb 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Do not Choose Representation just Change: An Experimental Study in States based EA

Maroun Bercachi Philippe Collard Manuel Clergue Sebastien Verel

I3S Laboratory - Nice-Sophia Antipolis University - CNRS
2000 Route Des Lucioles - 06903 Sophia Antipolis - France
{bercachi, pc, clergue, verel}@i3s.unice.fr, http://www.i3s.unice.fr/tea

ABSTRACT

Our aim in this paper is to analyse the phenotypic effects (evolvability) of diverse coding conversion operators in an instance of the states based evolutionary algorithm (SEA). Since the representation of solutions or the selection of the best encoding during the optimization process has been proved to be very important for the efficiency of evolutionary algorithms (EAs), we will discuss a strategy of coupling more than one representation and different procedures of conversion from one coding to another during the search. Elsewhere, some EAs try to use multiple representations (SM-GA, SEA, etc.) in intention to benefit from the characteristics of each of them. In spite of those results, this paper shows that the change of the representation is also a crucial approach to take into consideration while attempting to increase the performances of such EAs. As a demonstrative example, we use a two states SEA (2-SEA) which has two identical search spaces but different coding conversion operators. The results show that the way of changing from one coding to another and not only the choice of the best representation nor the representation itself is very advantageous and must be taken into account in order to well-desing and improve EAs execution.

Categories and Subject Descriptors

G.1.6 [**Mathematics of Computing**]: Numerical Analysis Optimization [Stochastic Programming]; I.2.8 [**Computing Methodologies**]: Artificial Intelligence Problem Solving, Control Methods, and Search [Heuristic Methods]

General Terms

Algorithms, Performance, Experimentation

Keywords

States based Evolutionary Algorithm, Representation, Coding Coupling, Coding Conversion.

1. INTRODUCTION

The choice of the representation of solutions is a very fundamental step and a highly decisive point to take into consideration in EAs functioning. A problem could be difficult for one representation and easy for another one [1, 2]. It is a challenging task to discover which coding scheme is a suitable one for a specific problem before testing that coding scheme using an evolutionary algorithm (EA). One representation could have a very good behaviour at the begining of the run and a bad one at the end of the run [3, 4, 2]. Besides, the search bias during genetic search depends on the problem, the structure of the encoded search space and the genetic operators of selection, crossover, and mutation. For every problem there is a large number of possible encodings. It is often possible to follow the principle of minimal alphabets when choosing an encoding for an EA, but simultaneously following the principle of meaningful building blocks can be much harder. This is because our intuition about the structure of the problem space may not translate well in the binary-encoded spaces that EAs expand and spread on [5, 6, 7, 8, 9]. There are two possible ways of tackling the problem of coding design for meaningful building blocks: 1) Search through possible encodings for a good one while searching for a solution and try to apply the chosen encoding. 2) Incorporate more than one coding scheme simultaneously and change the representation of solutions from one coding to another during the optimization process which can help in well exploring the search space and in increasing the count of building blocks considered meaningful in the solution string. The first choice uses reordering operators, like inversion, that try to look for and then apply the best encoding while searching for the solution. The rest of this paper motivates, develops and illustrates the second approach in use with the states based evolutionary algorithm (SEA).

The SEA is a new parallel version of EAs implemented as a group of independent optimization algorithms where each algorithm is considered as a state of a SEA. A state i of a SEA is denoted EA_i and can be any of the optimization algorithms that have been proposed in the literature such as EAs, genetic algorithms (GAs), genetic programming (GP), evolution strategies (ES), etc. An execution of a SEA with n states is equivalent to the execution of n parallel EAs where each EA_i has its own parameter settings (cf. Figure 1). After each main generation, a SEA contains a $merge$ phase which consists in regrouping all states together in a whole population. During this phase, each state undergoes a mutation to any other state with a given state mutation rate $pMutState$ which can help to maintain diversity over the state space. During the mutation phase, the states of existing solutions are converted without changing their corresponding fitness values. After the $merge$ phase, $selection\ for\ replacement$ and $elitist\ selection$ phases take place in order to guarantee the survival of the best individuals. The $elitist\ selection$ stage is done in the whole population according to the fitness values of each state. Finally, a $split$ phase

is necessary to disconnect all members of the whole population and reorder them in such a way that each homogeneous group reconstitutes a separate state. The $split$ and $merge$ cycle continues after each generation until the SEA obtains the ultimate solution or until a definite number of iterations is "absorbed" [10]. The main principle of the SEA is to choose the good state according to the fitness values of the actual solutions and not directly according to their states using a classical selection operator. Eventually, the SEA favours the coding whose the solutions have a best average fitness, and the choice of the coding depends on the evolvability of that coding just after the modification of representation. In another terms, it depends on the evolvability of the coding conversion operator which can be resumed as its capacity to promote and support the crossover and mutation operators to build new promising solutions from the old ones. Consequently, the conception of coding conversion operators that can lead to a scaled average fitness and an assorted evolvability would be a good approach to attack and dissolve the EAs problem caused by the representation issues.

On the other side, redundant representations are increasingly being applied in evolutionary computation and seem to affect positively the performance of genetic and evolutionary algorithms [7]. They use a higher number of alleles for encoding phenotypic information in the genotype than is essentially to construct the phenotype. This is the reason why we preface the use of block composites into the binary encoding which proves to have the features and properties of maintaining scaled genotypes and phenotypes (cf. Section 2). In this paper, we expose an already-evoked structure of binary representation tagged as binary block coding (BBC) and founded on the concept of bitstrings decomposed into a definite number of blocks each having a fixed length [7]. The previous work in [7] discussed how the synonymy of a representation influences the genetic search. Then, it developed a population sizing model for synonymously redundant representations based on the assumption that a representation affects the initial supply. Our present study will focus on the framework of coupling more than one representation in one algorithm. Eventually, this paper will be centralized on the concept of proposing diverse ways of coding conversion and on the matter of how and when to apply these conversion operators which allow to change the representation of solutions in the population from one coding to another. In this intention, two different ways of changing the representation are evolved in this paper. The first one tends to increase the number of zeros "$0s$" in the binary solution by the fact that each bit will be encoded as a sub-solution composed by a block of binary bits having the maximum number of "$0s$". The second way tends to increase the number of ones "$1s$" in the binary solution by the fact that each bit in the solution will be encoded as a sub-solution composed by a block of binary bits having the maximum number of "$1s$". Thus, the conversion operators are different but the representations used to encode individuals in the population are identical by the fact that they have the same search space, the same neighborhood structure and the same fitness values for individuals having equivalent solutions.

The experiments are performed in intention to prove that even if the representations used to encode the solutions are identical, the manner of modifying the representation from one coding to another in the algorithm is of importance and is useful in the same way of selecting the best representation. This paper contains four main sections. In Section 2, BBC is described in details. The entire set of experiments is exposed in Section 3. Section 4 presents general comments and concluding remarks. Finally, Section 5 summarizes some further works.

2. BBC AND CONVERSION

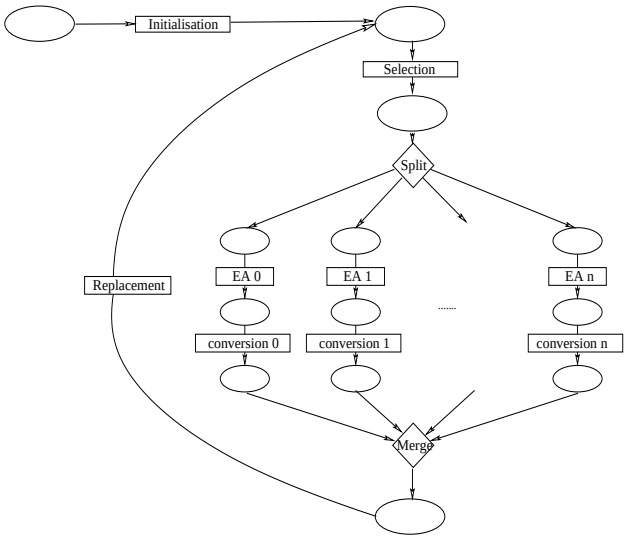


Figure 1: Scheme of a n-SEA.

The binary block coding and various coding conversion operators are evolved in this paper for the purpose of bringing some form of order into the disturbed situation caused by the influence of representation on the performance of EAs. The required specifications are outlined in the following subsections.

2.1 Binary Block Coding

We present the binary block coding scheme, an existing binary representation which is based on the binary block constitution [7]. BBC is the set of all possible solutions $\{0,1\}^{nk}$ where n is the blocks number and k is the block size. Suppose that we have a bitstring w which is encoded with BBC. w will be composed of a set of binary blocks w_j where $j \in [0, n-1]$ and each w_j is of length k (cf. Figure 2). The decoding of w to the standard binary returns a bitstring x of length n. This procedure can be defined by the binary voting mapping. Each block in w will be replaced by one bit in x. The value of each bit in x is determinded by the "voting to the majority of the values" in the corresponding block. Generally, a specific binary encoded optimization task requires a binary representation that correlates to its fitness function structure. In this intention and since a bit value can be set equal to "0" or "1", we state two variants of changing the binary representation from one coding to another. The first is assigned to maximize the number of "$0s$" in the bitstring by transforming each bit in the string to a block of binary substring containing the largest possible number of "$0s$" (cf. Figure 2). The second variant is assigned to maximize the number of "$1s$" in the bitstring by transforming each bit in the string to a block of binary substring containing the largest possible number of "$1s$" (cf. Figure 2). BBC is considered to introduce a form of redundancy to the chromosome codification and is itself an infinite group of binary coding schemes by just varying the block size, and the standard binary coding is the basic element of this group with a block size equal to 1.

2.2 BBC Encoding Operators

Suppose that we have a bitstring x of length n whose we want to encode in BBC with a block size equal to k generating as well a new bitstring w. As it has been mentioned above, two encoding operators are available to change the representation from standard

binary coding to BBC. The first, enc_0, maximizes the number of "$0s$" and the second, enc_1, maximizes the number of "$1s$" in the bitstring. So $\forall\, i \in \{0,1\}$, enc_i operator can be defined as follows:

$$enc_i : \{0,1\}^n \rightarrow \{0,1\}^{nk}$$

$$enc_i(x) = w = w_0 w_1 ... w_{n-1}$$

where $\forall\, j \in [0, n-1]$,

$$w_j = \begin{cases} i^k & \text{if } x_j = i \\ i^{\frac{k-1}{2}} \bar{i}^{\frac{k+1}{2}} & \text{if } x_j = \bar{i} \end{cases}$$

where $\bar{i}$ is the bitwise complement of i.
Two demonstrative examples are given in Figure 2.

2.3 BBC Decoding Operator

Suppose that we have a bitstring w composed of n blocks each having a size equal to k. If we want to decode w in standard binary, a new bitstring x will be generated using the decoding operator dec. The decoding procedure from BBC to standard binary coding is based on a predefined function called maj used to evaluate each w_j in w where $j \in [0, n-1]$. maj routine is specified by the "voting to the majority of the values" in the bitstring and it can be outlined as follows:

$$maj : \{0,1\}^k \rightarrow \{0,1\}$$

$$maj(u) = \begin{cases} 0 & \text{if } |u|_0 > |u|_1 \\ 1 & \text{otherwise} \end{cases}$$

where $|u|_0$ respectively $|u|_1$ represents the number of "$0s$" respectively of "$1s$" in u. Then, dec operator can be defined as follows:

$$dec : \{0,1\}^{nk} \rightarrow \{0,1\}^n$$

$$dec(w) = x = x_0 x_1 ... x_{n-1}$$

where $\forall\, j \in [0, n-1]$,

$$x_j = maj(w_j)$$

Two demonstrative examples are given in Figure 2.

2.4 BBC Conversion Operators

There exist several ways to change the representation of individuals in the population from BBC to BBC. We state below a brief list of two BBC conversion operators. They are transmutation utilities that change the state, here the representation, of a solution without changing the fitness value of that solution. So, we have $\forall\, j \in S$, $\forall\, x \in \Omega$, $f(x) = f(conv_j(x))$ where S is the state space, Ω is the search space, f the fitness function, and $conv_j$ the conversion operator to state j. In our case, $conv_0$ corresponds to the operator that maximizes the number of "$0s$" and $conv_1$ corresponds to the operator that maximizes the number of "$1s$" in the bitstring. Suppose that we have a bitstring w encoded with BBC and composed of n blocks each having a size equal to k. If we want to change the representation of w to BBC in a form of redundancy that increases the number of "is" in the bitstring with the same block size where $i \in \{0,1\}$, a new bitstring w' will be generated following two main steps. The first belongs to the decoding of w in standard binary producing as well a new bitstring x of length n. The second step belongs to the encoding of x in BBC by applying enc_i operator poducing as well a new bitstring w' of length nk. Therefore $\forall\, i \in \{0,1\}$, $conv_i$ operator can be defined as follows:

$$conv_i : \{0,1\}^{nk} \rightarrow \{0,1\}^{nk}$$

$$conv_i(w) = w' = enc_i(dec(w))$$

Two demonstrative examples are given in Figure 2.

2.4.1 Role and Importance

Some classes of optimization problems can take advantage from the coexistence and the application of the two BBC conversion operators, $conv_0$ and $conv_1$, in one algorithm. A dual coding strategy based on these two variants and developed genuinely in an EA serves to make the representation of solutions more adaptive and well-matched to a problem's fitness function. Likewise, this approach can make EAs advantageously explore undiscovered areas of the search space. If we introduce the notion of state to be defined according to the representation in a SEA, then that SEA can be the appropriate algorithm that integrates an adaptive approach for the representation in which the genotype encoding is altered dynamically by the fact that a state mutation will be equivalent to a coding conversion. Therefore, the modification of the representation of arbitrary solutions to a form of BBC using $conv_0$ or $conv_1$ tries to make an equilibrum in the number of bits with "0" and "1" while a classic binary representation sometimes makes bias towards the bits with "0" or "1". For example, if the ultimate solution of an optimization problem contains a number of "$0s$" more than the number of "$1s$" in the string then the BBC coding alternation "tour" performed in a SEA while applying $conv_0$ and then $conv_1$ to random solutions during the search may be helpful in increasing the number of bits with "0" and then can lead, iteration after iteration, to discover and locate the global optimum. The role of BBC conversion operators can be seen as intermediators between the standard binary coding and the problem structure, and those mediators serve to well explore new regions in the search space. The importance of those operators lies on the concept that specifies them as adjustors which attempt to correct the erroneous bits in the string by replacing each probable false bit value by the true one, the matter which can be seen and interpreted indirectly as the constructors of the meaningful building blocks. Since in a binary coding, "0" is the bitwise complement of "1" and inversely "1" is the bitwise complement of "0", so $conv_0$ can be translated as the complementary conversion operator of $conv_1$ and reciprocally $conv_1$ can be translated as the complementary conversion operator of $conv_0$. In this aim, we must notice that the value of BBC resides in using $conv_0$ and $conv_1$ operators simultaneously in one method that let them interact and interchange data bits to finally assisst in creating and not in destroying the substantive building blocks.

2.4.2 Evolvability

The evolvability of a coding conversion operator is defined as the phenotypic effects that can be produced after the change of the representation of solutions using that operator. In another terms, it is the ability of that operator to affect and serve the genetic operators, crossover and mutation, to develop new promising solutions from the old ones during the reproduction phase. Particularly, the evolvability of a coding conversion operator deeply depends on the problem structure and the shape of the optimum. Suppose that we have to optimize a problem where the global optimum contains a number of ones "1" greater than that of zeros "0". Then, $conv_1$ operator will be more favored regarding its concern in maximizing the number of ones "1" in the bitstring. Consequently, the chance to produce new promising solutions after the application of $conv_1$ will be greater than that after the application of $conv_0$, and hence the evolvability of $conv_1$ will be greater than that of $conv_0$. Two experimental tests were performed in sections 3.4.1 and 4 to study and compare the evolvability of $conv_0$ and $conv_1$ operators.

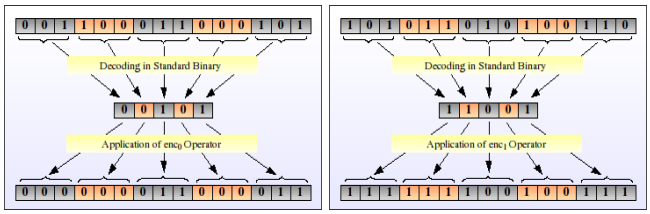
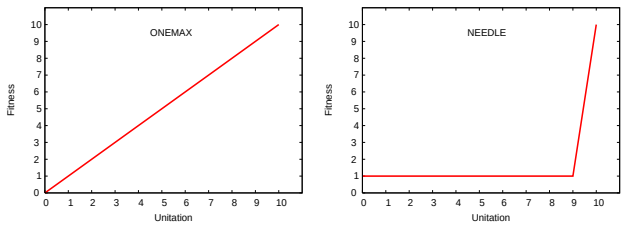


Figure 2: For two given bitstrings w and w' with a blocks number equal to 5 and a block size equal to 3 for both of them, we show the decoding in standard binary and then the application of enc_0 respectively of enc_1 operators.



ONEMAX NEEDLE

Figure 3: Graphical representations of unitation functions.

Table 1: Test Functions

Reference	Name	Definition
P1	ONEMAX	$f_1(s) = \|s\|_1$
P2	NEEDLE	$f_2(s) = \begin{cases} l & \text{if } \|s\|_1 = l \\ 1 & \text{otherwise} \end{cases}$
P3	ONOFF	$f_3(s) = HDO(s)$
P4	ALTERNATION	$f_4(s) = ND(s)$

3. EXPERIMENTS

We have prepared a set of experiments to test and analyze some of the main features of BBC conversion operators in use with a SEA, and to show the importance of changing the representation of solutions during the search process.

3.1 Test Functions

To test the performance of optimization algorithms, standard test problems should be used. We mainly consider a set of four binary encoded optimization functions.

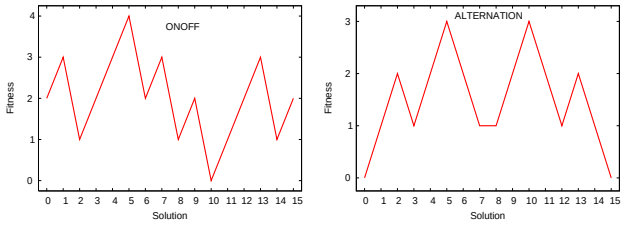
The first one is P1 and is the classical ONEMAX problem. It belongs to the unitation class of fitness functions. Unitation functions are fitness functions where the fitness is a function of the count of "$1s$" in a solution $x \in \{1,0\}^l$, where l is the length of the solution. All fitness values are non-negative: $u : \{0,1\}^l \to \mathbb{R}^+$. The first two fitness functions given in Table 1 and pictured in Figure 3 are two examples of unitation functions. They are respectively called ONEMAX and NEEDLE, and have been theoretically studied for fixed parameter simple GAs by Rowe [11], Wright [12] and Richter et al. [13]. The ONEMAX fitness function has been called the "fruit fly" of GA research [14]. It is a maximization problem that countes the number of "$1s$" in the string. P1 is a neutral linear function with one global optimum, an all "$1s$" string.

As well, we have expanded our observations to test the second function P2. It is the NEEDLE problem which also belongs to the unitation class of fitness functions. P2 has one global optimum, an all "$1s$" string, and is reasoned to be a difficult optimization task for the classic GA to work out. NEEDLE is a maximization linear problem and can serve to study the properties of the SEA and show the importance of changing the representation.

On the other side, we have applied our tests on the ONOFF problem P3. We define the ONOFF problem as a fitness function where the global optimum is a finite binary sequence of the form $1010...10$ and the fitness is the regular Hamming distance of a solution $x \in \{1,0\}^l$ to the global optimum, where l is the length of x. ONOFF is a typical minimization problem. All fitness values are non-negative: $u : \{0,1\}^l \to \mathbb{R}^+$, and the fitness value of the global optimum corresponds to a value of 0 for any length of the solution. Each bit of value 1 in the binary string of the global optimum represents the ON label and each bit of value 0 represents the OFF label. An illustrative example of the ONOFF function is pictured in Figure 4 for a length of the binary solution equal to 4. This function should advantageously confirm our assertions about changing the representation because we consider that the genuine solution of the form $1010...10$ will be a really challenging task for BBC conversion operators. Consequently, $conv_0$ and $conv_1$ operators should have the equal opportunities to be applied during the optimization task regarding the global optimum that contains an equal and consecutive number of "$0s$" and "$1s$".

Likewise, the experiments are extended to include the ALTERNATION problem P4. This function counts the number of dicontinuities between consecutive bits in the bitstring [15]. It is considered as a hard maximization problem for a simple GA to solve. It depends on the total number of sequences 10 or 01 in a string and not on the positions of the alternations. So, it is defined on the binomial distribution of the space induced by alternations. As a consequence, ALTERNATION function has the following properties: 1) All the points with the same number of alternations have the same fitness value. 2) Symmetry with respect to bit value, that is $f(x) = f(\bar{x})$, where $x \in \{1,0\}^l$ is a bitstring of length l, and $\bar{x}$ is its bitwise complement. 3) According to the above property, the fitness Hamming distance correlation coefficient is equal to zero. An illustrative example of the ALTERNATION function is pictured in Figure 4 for a length of the binary solution equal to 4. This problem provides an interesting tool to analyze and report the dimensions of enc_0 and enc_1 operators by the fact that it features two global optimum regarding its symmetry characteristic. The first is of the shape $1010...10$ and the second one is of the shape $0101...01$. Contrarily to the first three problems, P4 is a non-linear problem where a form of epistasis is contained in the structure of the solution and the bits are tightly linked each to other. The chances to apply $conv_0$ and $conv_1$ operators must be equivalent for the EA to succeed.

The definitions of all these problems are summarized in Table 1 where l is the length of the solution s, $|s|_1$ is the number of "$1s$" in s, $HDO(s)$ is the Hamming distance of s to the global optimum, and $ND(s)$ is the count of dicontinuities between consecutive bits in s. In order to compute the fitness value f' of a given solution w which is encoded by BBC where $f' : \{0,1\}^{nk} \to \mathbb{R}^+$, first we decode w in standard binary generating as well a new bitstring x. And then, the fitness value f of x is taken equal to the corresponding function value which is calculated according to the function expression given in Table 1 where $f : \{0,1\}^n \to \mathbb{R}^+$. And so, we obtain the following equality: $f' = f \circ dec$.



ONOFF ALTERNATION

Figure 4: Examples of the graphical representation of the ONOFF and ALTERNATION functions. For a binary solution of length equal to 4, we compute the fitness values corresponding respectively to the 16 (2^4) possible solutions. The x-axis represents the real-value of each standard binary solution. The y-axis represents the fitness of solutions.

3.2 SEA Parameters

Since a SEA itself has several options in terms of its implementation, it is necessary to denote the parameter choices used in this paper. First, the number of states was set to 2, each state being represented by a simple GA (SGA). The representations of solutions applied in all states are identical and we used BBC for encoding the solutions. This choice was explicit in intention to integrate a dual coding strategy which may help in locating the ultimate solution while changing the representation of different random solutions from one coding to another using BBC conversion operators. Thus, we have two states symbolized by two SGAs which are similar in everything and each component. And so, an instance of the SEA is implemented and is denoted 2-SEA where the representation is directly linked to the algorithm and not to the individual in the population. Each SGA is executed for one simple iteration before the $merge$ phase takes place in the algorithm life-cycle. Next, the follow-up parameter is $pMutState$ for state mutation rate. In our case and since the representation is directly linked to the algorithm, a state mutation means that the representation of individuals in that algorithm is changed to another representation. Afterwards, we will refer to $pMutState$ by conversion rate. $pMutState$ parameter could be easily modified to provide conversion of arbitrary solutions from one coding scheme to another without affecting the results dramatically during the search process. The value of this parameter is fixed using the experiments described later in Section 3.4.1. As well, 2-SEA has another particular parameter: k for the block size. The value used for that parameter was chosen as a result of prior experimentation reported subsequently in Section 3.4.1. The best parameter settings between those tested for all objective functions are given in Table 3.

3.3 General Parameter Values

In order to create a fair tableau for comparison of SGA with 2-SEA, the parameters shared between these two algorithms were kept the same. Since 2-SEA is composed of two parallel SGAs, the classic GA and 2-SEA were run with the parameters recommended by Goldberg (Goldberg 1989) (cf. Table 2). In general, the set of all used parameters and their respective attributes are shown in Table 3 with: $maxGen$ for maximum number of generations before STOP, $popSize$ for population size, $vecSize = nk$ for genotype size, $tSize$ for tournament selection size, $pCross$ for crossover rate, $pMut$ for mutation rate, and $pMutPerBit$ for bit-flip mutation rate. This tableau was employed for the four test problems. We have to mention that for the first three test problems the population

Table 2: General Parameter Values

Parameters	Attributes
$Pseudorandom\ generator$	Uniform Generator
$Selection\ mechanism$	Tournament Selection
$Crossover\ mechanism$	1-Point Crossover
$Mutation\ mechanism$	Bit-Flip Mutation
$Replacement\ model\ 1$	Generational Replacement
$Replacement\ model\ 2$	Elitism Replacement
$Ending\ criteria$	Maximum Number of Iterations

Table 3: Best Parameter Settings

Parameters	P1	P2	P3	P4
$maxGen$	3000	3000	3000	30000
$popSize$	100	100	100	10
$vecSize$	1900	1900	300	300
$tSize$	2	2	2	2
$pCross$	0.6	0.6	0.6	0.6
$pMut$	1.0	1.0	1.0	1.0
$pMutPerBit$	0.9	0.9	0.05	0.05
$pMutState$	1.0	1.0	0.85	0.7
k	19	19	3	3

size was set equal to 100 and for the last problem this parameter value was set equal to 10 which reflects the fact that the ALTERNATION function requires more exploitation than exploration due to the deceptive attractor which is at mid-distance from the global optimum. This choice is well verified and is totally compatible with the choice of a low $pMutPerBit$ value for the ALTERNATION function which enables the algorithm to discover recursively and regularly good directions in the search interval.

3.4 Experimental Results

In the following two subsections, we introduce the experiments that have been performed for two different purposes. The first serves to analyze BBC conversion operators and to study the interaction and the dependency of the parameters of both BBC and 2-SEA. And the second purpose tries to test the importance of changing the representation and contributes in a comparison between the performance of 2-SEA and the classic GA.

3.4.1 BBC Analysis

In this section, we present experiments designed to examine several aspects of BBC conversion operators. We would like to know how much the change of the representation using $conv_0$ and $conv_1$ could "help" and "advance" 2-SEA during the search. Besides, we would like to discover how the parameters of both BBC and 2-SEA interact each with other.

First, it is so essential to mention that $pMutState$ and $pMutPerBit$ parameters play an important role in 2-SEA operation, and their affected values are decisive in the final outcome. Precisely, $pMutState$ is responsible for the conversion of arbitrary individuals in the population from their initial representation to the other one. In our research, we are using $conv_0$ and $conv_1$ as two different conversion operators for the same search space defined by BBC. Consequently, each of these two operators has a different evolvability after the change of the representation. Since the evolvability of a coding conversion operator and with it $pMutState$ is incidental to

the application of genetic operators and with it to the probability of flipping one bit in a bitstring, $pMutPerBit$, we will begin by exploring the relationship between $pMutState$ and $pMutPerBit$ parameters and the proportion of solutions solved correctly by 2-SEA, success rate in percent. In the first experiment, $pMutState$ and $pMutPerBit$ values changed within $[0.0 : 1.0]$ interval with a step of 0.05. This experiment was realized on each test problem for 100 independent runs. Graphical representations of fitness variations relatively to $pMutState$ and $pMutPerBit$ were given in Figure 5. A simple reading of these figures shows that a large conversion rate is needed for all test functions in order for 2-SEA to produce positive results which reflects the great importance and utility of the change of the representation during the search. Besides, Figure 5 indicates that a high bit-flip mutation rate is required for P1 and P2 problems, and a small bit-flip mutation rate is required for P3 and P4 problems so that 2-SEA can render important end results. As an elementary synthesis on these obtained results, we can say that P3 and P4 problems require a low-level of mutation effects regarding the ordered structure of their global optimums which necessitate a modest contribution of the genetic operators, especially the bit-flip mutation, to be able to rearrange and fix up each bit in its correct position in the bitstring.
On the other side, BBC has another key parameter: k for the block size. The second experiment is performed to determine the value of that parameter for each test function. First, we have fixed the length of the standard binary genotype to a value of $n = 100$ which means that the number of blocks in the binary block genotype will be equal to 100 and the fitness value of the global optimum will be equal to 100 for unitation functions, 0 for P3 problem, and 99 for P4 problem. Likewise, we have fixed the values of $pMutState$ and $pMutPerBit$ parameters respectively to 1.0 and 0.9 for unitation functions. P3 and P4 problems have a $pMutState$ value equal to 0.85 respectively 0.7 and a $pMutPerBit$ value equal to 0.05 for both of them. This test was realized on all objective functions for 100 independent runs. Graphical records are displayed in Figure 6 and show that a large block size is necessary for the unitation functions in order for 2-SEA to produce significant positive results in a minimum number of iterations. This fact can be explained as a consequence of that, for classical linear problems, an optimal evolvability of a coding conversion operator is related to a maximal length of a bitstring and next to a maximal or large block size. On the other side, P3 and P4 problems require a small block size to make 2-SEA competent to submit large-scale solutions by the concept of that a small $pMutPerBit$ value and with it a low-order evolvability can avoid a disruptive effect on the solution and as a result it can help in adjusting the structural form of individuals heuristically and progressively in a minor number of generations.

3.4.2 Performance Comparison: 2-SEA vs. sGA

Considering the stochastic nature of 2-SEA, we compute the average performance of 100 independent runs of 2-SEA on each objective function. The global optimum being equal to 100 for P1 and P2, 0 for P3 and 99 for P4, Table 4 shows the numerical results whereas Figure 8 represents the graphical records of the experiments. For the two algorithms, SGA and 2-SEA, Table 4 displays two main records for each test function. The first is the success rate ($SR_{\%}$) measurement and is the percentage of the number of runs in which the algorithm succeeded in finding the global optimum. The second record is the generation number to optimum (GNTO) measurement and is the average of the number of iterations needed for the algorithm to attain the global optimum.

4. DISCUSSION AND CONCLUSION

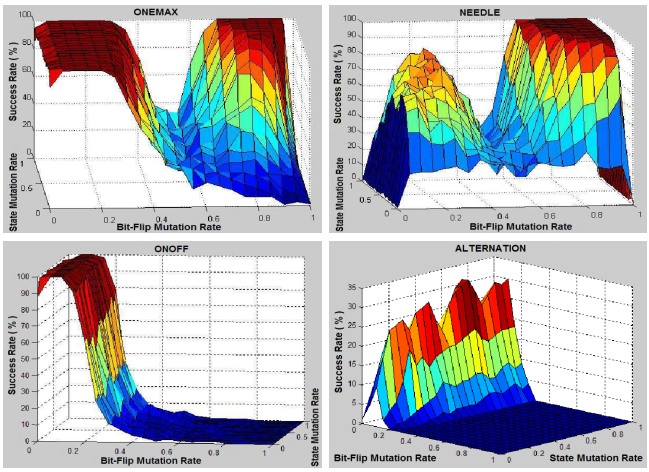


Figure 5: Study of the success rate fluctuations relatively to the variations of the conversion rate and the bit-flip mutation rate. $pMutState$ and $pMutPerBit$ values varied from 0.0 to 1.0 with a step of 0.05 for all test functions. As a result, the obtained success rate values varied from 0% to 100%.

Table 4: Experimental Results

Problem	Measurement	Algorithm	
		SGA	2-SEA
P1	SR %	100	100
	GNTO	128	10
P2	SR %	3	100
	GNTO	3000+	8
P3	SR %	100	100
	GNTO	579	84
P4	SR %	4	32
	GNTO	30000+	30000+

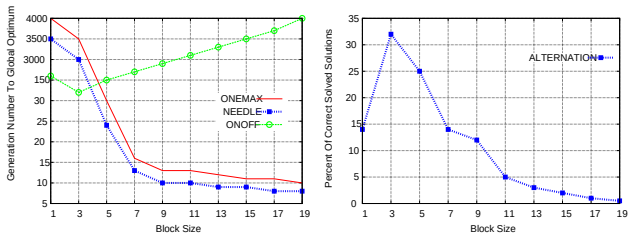


Figure 6: For the best values of $pMutState$ and $pMutPerBit$, we plot the number of iterations required to reach the global optimum relatively to the block size for P1, P2 and P3 problems. As well, we plot the percent of correct solved solutions relatively to the block size for P4 problem.

In this paper, we used a method based on the framework of change of the representation during the search. The basic intermediators to apply this action were the diverse BBC conversion operators. These operators allow to alter the representation of solutions from one coding to another during the search without modifying their respective fitness values. For this purpose, we applied identical representations of BBC which have the same search space, the same neighborhood structure and the same fitness values for identical solutions, but various conversion operators to change the form of the representation issued from BBC. We have to state that all previous works which used to search for the good coding during the optimization process and then tried to apply that best coding were very essential, helpful and efficient. Another effective statement and affirmation can be deduced from our work and test results. The data of the experiments shown in Table 4 and Figure 8 clearly prove the importance and the utility of changing the representation of many random solutions in favour of 2-SEA that incorporates a conversion strategy which leads to a dynamic and mutual representation. They also confirm that the change of the representation during the search is a very helpful and fundamental step to profoundly think about as well to apply when one tries to improve EAs performances.

Besides, the experimental results displayed in Table 4 and Figure 8 are uncomparable and show the advancement of 2-SEA over SGA. They distinctly show how 2-SEA has found the global optimum in an extreme minimal number of generations for the first three test functions while SGA has reached the global optimum of the ONEMAX and ONOFF problems in a remarkable larger number of iterations and failed to detect the global optimum of the NEEDLE and ALTERNATION problems for the majority of runs (for a great proportion of initial populations). For the deceptive P4 problem, the change of the representation with an appropriate conversion rate has driven the search process in 2-SEA to build and fix each bit in its correct position relatively to its neighbours but good combinations of bits cannot be made fast enough because of the matter that the bits are tightly linked each to other. Consequently, the change of the representation during the optimization task has been proved to be of great importance in EAs operation and positively showed that the obtained results for 2-SEA are significantly different from those of SGA for all objective functions.

To prove our results, first we must show value of applying the coding conversion operators. Thus, to reveal some characteristics of BBC and study the evolvability of $conv_0$ and $conv_1$, we have made a simple test on the ONEMAX problem denoted *Fitness Clouds Representation* [16]. We started our test with a fixed number of arbitrary solutions uniformly generated from a given $seed$ number. In a first step, we applied a standard bit-flip mutation to each of those solutions and evaluated their respective fitnesses (m). As a next step, we applied two kinds of coding conversion to those initial solutions, the first is done using $conv_1$ operator and the second using $conv_0$ operator. Then, we applied a standard bit-flip mutation to each of those solutions and evaluated their respective fitnesses ($m \circ conv_1$ and $m \circ conv_0$). In a following step, we applied two types of coding alternation "tour" to the same random solutions taken before, where each "tour" is considered as two consecutive coding conversions. The first "tour" is realized according to the respective application of $conv_1$, a bit-flip mutation, $conv_0$, a bit-flip mutation, and the evaluation of the corresponding fitnesses ($m \circ conv_1 \circ m \circ conv_0$). Inversely, the second "tour" is realized according to the respective application of $conv_0$, a bit-flip mutation, $conv_1$, a bit-flip mutation, and the evaluation of the corresponding fitnesses ($m \circ conv_0 \circ m \circ conv_1$). This elementary test was performed on 100 arbitrary solutions, each having a length of 1900, an extreme value of k equal to 19, a blocks number equal to 100, and hence the ultimate solution is an all "$1s$" string with a fitness value equal to 100. The traditional bit-flip mutation operator was applied in all steps with a bit-flip mutation rate equal to 0.25. The comparison of the obtained results is illustrated in Figure 7.

A graphical interpretation of Figure 7 (left) indicates that the fitness values of individuals which have been submitted to $conv_1$ and then to the bit-flip mutation operator are higher than those of individuals which have been simply submitted to a bit-flip mutation and than those of individuals which have been submitted to $conv_0$ and then to the bit-flip mutation operator. Similarly, Figure 7 (right) shows that the fitness values of individuals which have been submitted to $conv_0$ and then to $conv_1$ and then to the bit-flip mutation operator are higher than those of individuals which have been simply submitted to a bit-flip mutation and than those of individuals which have been submitted to $conv_1$ and then to $conv_0$ and then to the bit-flip mutation operator. Since for the ONEMAX problem, the more the number of "$1s$" in the string increases the more the corresponding fitness value increases, Figure 7 proves very well that $conv_1$ is the most appropriate conversion operator and is the one that clearly contributed in producing superior results. We can conclude that the coding alternation "tour" and the conversion of the representation from one coding to another have induced a befitting evolvability that matches to the problem structure. And the test results assume that the last applied BBC conversion operator is the one that influences the more on the final outcome. Graphical representations of fitness variations relatively to $pMutState$ given in Figure 5 showed that, for all test functions, a large conversion rate is needed for 2-SEA to render high positive results, the fact that justifies once a time the important and essential role of BBC coding alternation "tour" and its constructive influence on the performance of 2-SEA by re-creating, remodeling and reforming the meaningful building blocks. Also, we can say that the evolvability of a coding conversion operator is more beneficial over the EAs performance after the change of the representation. The use of more than one coding in EAs is very important and the most fundamental mechanism resides in the framework of changing the representation from one coding to another which contributes in exploring undetected and unspoiled sub-regions of the search space. Therefore, better fitness values can be discovered and EAs can progress towards more positive outcomes. The test results have distinctly verified the utility of BBC and the value of coupling various encodings in an alternation strategy where different conversion operators interact to increase the probabilities of obtaining advanced and good structures.

As a final statement, we believe that our test results totally complied with our assertions about the argument "Do not Choose Representation just Change", and showed that an algorithm which incorporates a method of coding mating by the application of the conversion of the representation from one coding to another during the search will easily converge and will be more successful in reaching more optimum solutions using less computational power.

5. FUTURE DIRECTIONS

In our study, we used two SGAs in 2-SEA. An advanced research can lead to the exploitation of other kinds of EAs to be assigned to each state having in mind that both the notion of states and the state conversions are very essential in EAs functioning.

In this paper, BBC is considered. Though, other kinds of coding schemes such as tree or linear representation, and any number of coding schemes can be applied to EAs in order to profit from the convenient representation for a particular problem.

A future direction also suggests that other implementations of the SEA can still be improved by decreasing user defined parameters

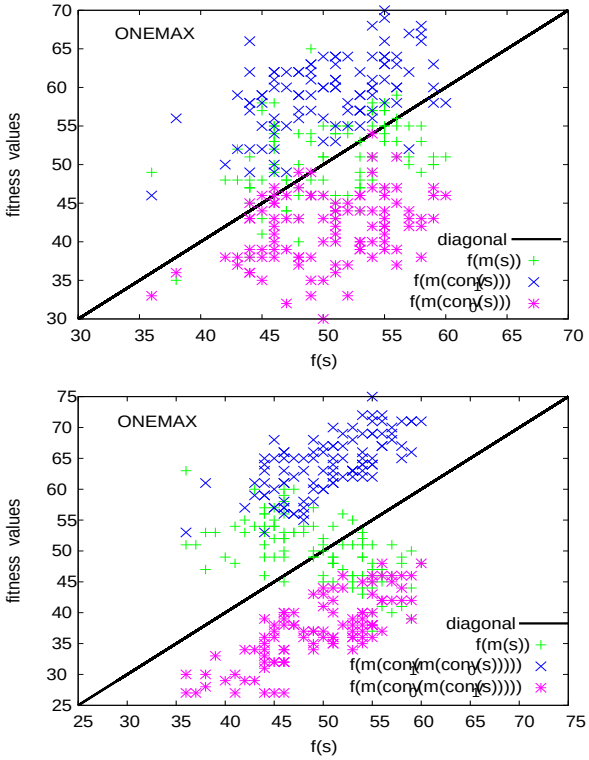


Figure 7: Fitness Clouds representing the influence and interaction of BBC conversion routines with genetic operators: for the ONEMAX problem, we plot different kinds of fitness values for 100 arbitrary solutions. The x-axis represents the fitness of initial solutions (f). The y-axis represents the fitness of solutions after the application of BBC conversion operators ($conv$) and the bit-flip mutation operator (m).

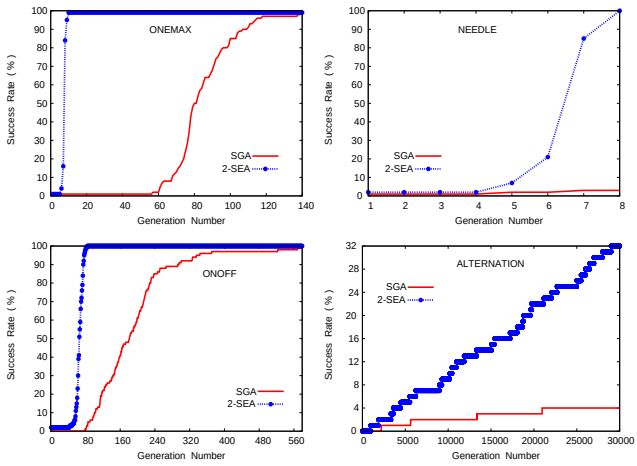


Figure 8: Performance comparison of the percentage of correct solved solutions across the number of iterations required to reach the global optimum. These records were averaged over 100 independent runs for each test function.

and making them automatically adjustable based on measures extracted from the process.

In a further research, we must understand properly the basic properties of BBC and recognize well its fundamental evolvability evoked by the genetic operators so we can propose other types of BBC conversion operators to help making the representation more dynamic and more adaptive to the problem structure.

6. REFERENCES

[1] D.H. Wolpert and W.G. MacReady. "No free lunch theorems for optimization". *IEEE Transactions on Evolutionary Computation*, 1996.

[2] Maroun Bercachi, Philippe Collard, Manuel Clergue, and Sebastien Verel. "Evolving dynamic change and exchange of genotype encoding in genetic algorithms for difficult optimization problems". In *Proceedings of IEEE International Congress on Evolutionary Computation*, 2007.

[3] M. Mitchell. "An introduction to genetic algorithms". *Complex Adaptive Systems Series*, MIT Press 1996.

[4] D. Whitley and Soraya Rana. "Representation, search and genetic algorithm". In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 1997.

[5] Franz Rothlauf and David E. Goldberg. "Representations for genetic and evolutionary algorithms". Springer 2002.

[6] Liepins G. and Vose M. "Representations issues in genetic algorithms". *Experimental and Theoretical Artificial Intelligence Journal*, MIT Press 1990.

[7] Franz Rothlauf and David E. Goldberg. "Redundant representations in evolutionary computation". *Evolutionary Computation*, MIT Press 2003.

[8] Marc Toussaint. "Compact representations as a search strategy: Compression edas". E.S. Publishers Ltd. 2006.

[9] Franz Rothlauf, David E. Goldberg, and Armin Heinzl. "Network random keys: a tree representations scheme for genetic and evolutionary algorithms". MIT Press 2002.

[10] Sebastien Verel. "States based evolutionary algorithm". *Internal report*, I3S Laboratory 2008.

[11] Rowe J.E. "Population fixed-points for functions of

unitation". In *Foundations of Genetic Algorithms. Vol. 5. Colin Reeves, and Wolfgang Banzhaf*. M.K. Publishers, 1998.

[12] Wright A.H., Rowe J.E., and Neil J.R. "Analysis of the simple genetic algorithm on the single-peak and double-peak landscapes". In *Proceedings of the 2002 Congress on Evolutionary Computation, Fogel et al. Editors*, 1999.

[13] Richter J.N., Paxton J., and Wright A.H. "EA models and population fixed points versus mutation for functions of unitation". In *Proceedings of GECCO 2005*, D.C. 2005.

[14] Thierens D. "Adaptive mutation rate control schemes in genetic algorithms". In *Proceedings of IEEE World Congress on Computational Intelligence: Congress on Evolutionary Computation*. IEEE Press, 2002.

[15] Philippe Collard and Manuel Clergue. "Misleading functions designed from alternation". In *Proceedings of IEEE International Congress on Evolutionary Computation*, 2000.

[16] Sebastien Verel, Philippe Collard, and Manuel Clergue. "Where are bottlenecks in nk fitness landscapes?". In *Proceedings of IEEE International Congress on Evolutionary Computation*, 2003.

[17] Rich Caruana, Schaffer, and J. David. "Representation and hidden bias: Gray vs. binary coding for genetic algorithms". In *Proceedings of the Fifth International Conference on Machine Learning*. M.K. Publishers, 1988.

[18] E. Mathias and D. Whitley. "Transforming the search space with gray coding". In *Proceedings of the 1994 International Conference on Evolutionary Computation*, 1994.

[19] D. Whitley. "A free lunch proof for gray versus binary encodings". In *Proceedings of the Genetic and Evolutionary Computation Conference*, M.K. Publishers 1999.

[20] D. Whitley, Soraya Rana, and Robert B. Heckendorn. "Representation issues in neighborhood search and evolutionary algorithms". *Genetic Algorithms in Engineering and Computer Science*, 1997.

[21] Laura Barbulescu, Jean-Paul Watson, and D. Whitley. "Dynamic representations and escaping local optima: Improving genetic algorithms and local search". 2000.