5

A Splicing/Decomposable Binary Encoding and Its Novel Operators for Genetic and Evolutionary Algorithms

Yong Liang
Macau University of Science and Technology
China

1. Introduction

Most of the real-world problems could be encoded by different representations, but genetic and evolutionary algorithms (GEAs) may not be able to successfully solve the problems based on their phenotypic representations, unless we use some problem-specific genetic operators. Therefore, a proper genetic representation is necessary when using GEAs on the real-world problems (Goldberg, 1989; Liepins, 1990; Whitley, 2000; Liang, 2011).

A large number of theoretical and empirical investigations on genetic representations were made over the last decades. Earlier work (Goldberg, 1989c; Liepins & Vose, 1990) has shown that the behavior and performance of GEAs is strongly influenced by the representation used. As a result many genotypic representations were made for proper GEAs searching. Among of them, the binary, integer, real-valued, messy and tree structure representations are the most important and widely used by many GEAs.

To investigate the performance of the genetic representations, originally, the schema theorem proposed by Holland (1975) to model the performance of GEAs to process similarities between binary bitstrings. Using the definition of the building blocks (BBs) as being highly fit solutions to sub-problems, which are decomposed by the overall problem, the building block hypothesis (Goldberg, 1989c) states that GEAs mainly work due to their ability to propagate short, low order and highly fit BBs. During the last decade, (Thierens, 1995; Miller, 1996; Harik, 1997; Sendhoff, 1997; Rothlauf, 2002) developed three important elements towards a general theory of genetic representations. They identified that redundancy, the scaling of Building Blocks (BBs) and the distance distortion are major factors that influence the performance of GEAs with different genetic representations.

A genetic representation is denoted to be redundant if the number of genotypes is higher than the number of phenotypes. Investigating redundant representation reveals that give more copies to high quality solutions in the initial population result in a higher performance of GEAs, whereas encodings where high quality solutions are underrepresented make a problem more difficult to solve. Uniform redundancy, however, has no influence on the performance of GEAs.

The order of scaling of a representation describes the different contribution of the BBs to the individual's fitness. It is well known that if the BBs are uniformly scaled, GEAs solve all BBs

implicitly in parallel. In contrast, for non-uniformly scaled BBs, domino convergence occurs and the BBs are solved sequentially starting with the most salient BB (Thierens, 1995). As a result, the convergence time increases and the performance is decreasing due to the noise from the competing BBs.

The distance distortion of a representation measures how much the distance between individuals are changed when mapping the phenotypes to the genotypes, and the locality of the representation means that whether similar genotypes correspond to similar phenotypes. The theoretical analysis shows that representation where the distance distortion and locality are equal to zero, that means the distances between the individuals are preserved, do not modify the difficulty of the problems they are used for, and guarantee to solve problems of bounded complexity reliably and predictably.

The importance of choosing proper representations for the performance of GAs is already recognized, but developing a general theory of representations is a formidable challenge. Up to now, there is no well set-up theory regarding the influence of representations on the performance of GAs. To help users with different tasks to search good representations, over the last few years, some researchers have made recommendations based on the existing theories. For example, Goldberg (Goldberg, 1989) proposed two basic design principles for encodings:

- Principle of minimal alphabets: The alphabet of the encoding should be as small as possible while still allowing a natural representation of solutions.
- Principle of meaningful building blocks: The schemata should be short, of low order, and relatively unrelated to schemata over other fixed positions.

The principle of minimal alphabets advises us to use bit string representation. Combining with the principle of meaningful building blocks (BBs), we construct uniform salient BBs, which include equal scaled and splicing/decomposable alleles.

The purpose of this chapter is to introduce our novel genetic representation — a splicing/decomposable (S/D) binary encoding, which was proposed based on some theoretical guidance and existing recommendations for designing efficient genetic representations. The S/D binary representation can be spliced and decomposed to describe potential solutions of the problem with different precisions by different number of uniform-salient BBs. According to the characteristics of the S/D binary representation, GEAs can be applied from the high scaled to the low scaled BBs sequentially to avoid the noise from the competing BBs and improve GEAs' performance. Our theoretical and empirical investigations reveal that the S/D binary representation is more proper than other existing binary encodings for GEAs searching. Moreover, a new genotypic distance d_g on the S/D binary space Φ_g is proposed, which is equivalent to the Euclidean distance d_p on the real-valued space Φ_p during GEAs convergence. Based on the new genotypic distance d_g, GEAs can reliably and predictably solve problems of bounded complexity and the methods depended on the phenotypic distance d_p for solving different kinds of optimization problems can be directly used on the S/D binary space Φ_g.

This chapter is organized as follows. Section 2 describes three most commonly used binary representations — binary, gray and unary encodings, and their theoretical analysis of the effect on the performance of GEAs. Section 3 introduces our proposed splicing/decomposable

(S/D) binary representation and its genotypic distance. Section 4 proposes the new genetic algorithm based on the S/D binary representation, the splicing/Decompocable genetic algorithm (SDGA). Section 5 discusses the performance of the SDGA and compares the S/D binary representation with other existing binary encodings from the empirical studies. The chapter conclusion are drawn in Section 6.

2. Background

Binary encodings are the most commonly used and nature-inspired representations for GEAs, especially for genetic algorithms (GAs) (Goldberg, 1989). When encoding real-valued problems by binary representations, different types of binary representations assign the real-value in different ways to the binary strings. The most common binary representations are the binary, gray and unary encodings. According to three aspects of representation theory (redundancy, scaled building block and distance distortion), Rothlauf (Rothlauf, 2002) studied the performance differences of GAs by different binary representations for real encoding.

2.1 The unary encoding and redundancy

In the unary encoding, a string of length $l = s - 1$ is necessary to represent s different phenotypic values. The i^{th} phenotypic value is encoded by the number of ones $i - 1$ in the corresponding genotypic string. Thus, 2^{s-1} different genotypes only encode s different phenotypes. Analysis on the unary encoding by the representation theory reveals that encoding is redundant, and does not represent phenotypes uniformly. Therefore, the performance of GAs with the unary encoding depends on the structure of the optimal solution. Unary GAs fail to solve integer one-max, deceptive trap and BinInt (Rothlauf, 2002) problems, unless larger population sizes are used, because the optimal solutions are strongly underrepresented for these three types of problems. Thus, the unary GAs perform much worse than GAs using the non-redundant binary or gray encoding (Julstrom, 1999; Rothlauf, 2002).

2.2 The binary encoding, scaled building blocks and hamming cliff

The binary encoding uses exponentially scaled bits to represent phenotypes. Each phenotypic value $x_p \in \Phi_p = \{x_1, x_2, ..., x_s\}$ is represented by a binary string x_g of length $l = log_2(s)$. Therefore, the genotype-phenotype mapping of the binary encoding is one-to-one mapping and encodes phenotypes redundancy-free.

However, for non-uniformly binary strings and competing Building Blocks (BBs) for high dimensional phenotype space, there are a lot of noise from the competing BBs lead to a reduction on the performance of GAs. The performance of GAs using the binary encoding is not only affected by the non-uniformly scaling of BBs, but also by problems associated with the *Hamming cliff* (Schaffer, 1989b). The binary encoding has the effect that genotypes of some phenotypical neighbors are completely different. For example, when we choose the phenotypes $x_p = 7$ and $y_p = 8$, both individuals have a distance of one, but the resulting genotypes $x_g = 0111$ and $y_g = 1000$ have the largest possible genotypic distance $\|x - y\|_g = 4$. As a result, the locality of the binary representation is partially low. In the distance distortion theory, an encoding preserves the difficulty of a problem if it has perfect locality and if it does not modify the distance between individuals. The analysis reveals that the binary encoding

changes the distance between the individuals and therefore changes the complexity of the optimization problem. Thus, easy problems can become difficult, and vice versa. The binary GAs are not able to reliably solve problems when mapping the phenotypes to the genotypes.

2.3 The gray encoding and modification of problem difficulty

The non-redundant gray encoding (Schaffer, 1989a) was designed to overcome the problems with the *Hamming cliff* of the binary encoding (Schaffer, 1989b). In the gray encoding, every neighbor of a phenotype is also a neighbor of the corresponding genotype. Therefore, the difficulty of a problem remains unchanged when using mutation-based search operators that only perform small step in the search space. As a result, easy problems and problems of bounded difficulty are easier to solve when using the mutation-based search with the gray coding than that with the binary encoding. Although the gray encoding has high locality, it still changes the distance correspondence between the individuals with bit difference of more than one. When focused on crossover-based search methods, the analysis of the average fitness of the schemata reveals that the gray encoding preserves building block complexity less than the binary encoding. Thus, a decrease in performance of gray-encoded GAs is unavoidable for some kind of problems (Whitley, 2000).

3. A novel splicing/decomposable binary genetic representation

The descriptions in above section show that the existing binary genetic representations are not proper for GAs searching and cannot guarantee that using GAs to solve problems of bounded complexity reliably and predictably. According to the theoretical analysis and recommendations for the design of an efficient representation, there are some important points that a genetic representation should try to respect. Common representations for GAs often encode the phenotypes by using a sequence of alleles. The alleles can separated (decomposed) into building blocks (BBs) which do not interact with each other and which determine one specific phenotypic property of the solution. The purpose of the genetic operators is to decompose the whole sequence of alleles by detecting which BBs influence each other. GAs perform well because they can identify best alleles of each BB and combine them to form high-quality over-all solution of the problem.

Based on above investigation results and recommendations, we have proposed a new genetic representation, which is proper for GAs searching. In this section, first we introduce a novel splicing/decomposable (S/D) binary encoding, then we define the new genotypic distance for the S/D encoding, finally we give the theoretical analysis for the S/D encoding based on the three elements of genetic representation theory (redundancy, scaled BBs and distance distortion).

3.1 A splicing/decomposable binary encoding

In (Leung, 2002; Xu, 2003a), we have proposed a novel S/D binary encoding for real-value encoding. Assuming the phenotypic domain Φ_p of the n dimensional problem can be specified by

$$\Phi_p = [\alpha_1, \beta_1] \times [\alpha_2, \beta_2] \times \cdots \times [\alpha_n, \beta_n].$$

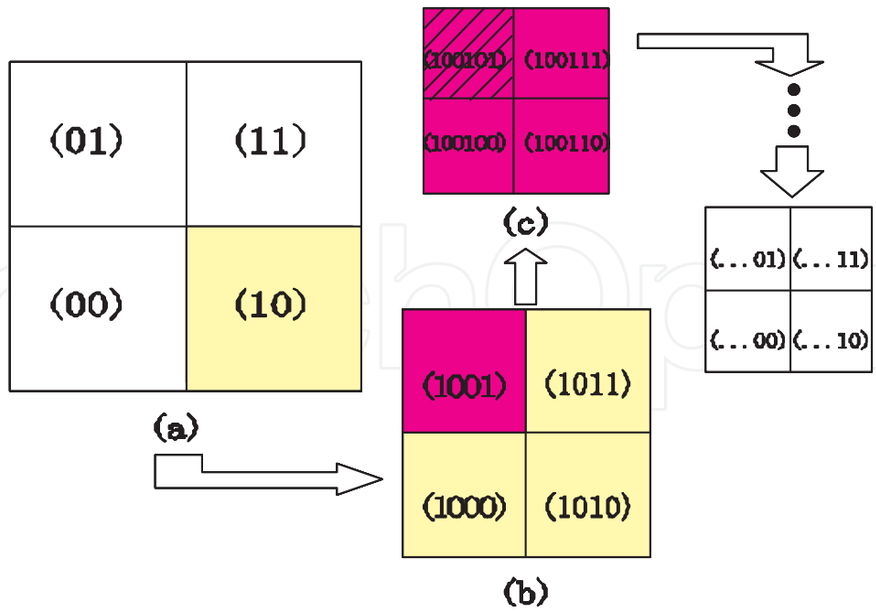


Fig. 1. A graphical illustration of the splicing/decomposable representation scheme, where (b) is the refined bisection of the gray cell (10) in (a) (with mesh size $O(1/2)$), (c) is the refined bisection of the dark cell (1001) in (b) (with mesh size $O(1/2^2)$), and so forth.

Given a length of a binary string l, the genotypic precision is $h_i(l) = \frac{(\beta_i - \alpha_i)}{2^{(\ell/n)}}$, $i = 1, 2, \cdots, n$. Any real-value variable $x = (x_1, x_2, ..., x_n) \in \Phi_p$ can be represented by a splicing/decomposable (S/D) binary string $b = (b_1, b_2, .., b_l)$, the genotype-phenotype mapping f_g is defined as

$$x = (x_1, x_2, \cdots, x_n) = f_g(b) = (\sum_{j=0}^{l/n} 2^{(l/n-j)} \times b_{j\times n+1},$$

$$\sum_{j=0}^{l/n} 2^{(l/n-j)} \times b_{j\times n+2}, \cdots, \sum_{j=0}^{l/n} 2^{(l/n-j)} \times b_{j\times (n+1)}),$$

where

$$\sum_{j=0}^{l/n} 2^{(l/n-j)} \times b_{j\times n+i} \leq \frac{x_i - \alpha_i}{h_i(l)} < \sum_{j=0}^{l/n} 2^{(l/n-j)} \times b_{j\times n+i} + 1.$$

That is, the significance of each bit of the encoding can be clearly and uniquely interpreted (hence, each BB of the encoded S/D binary string has a specific meaning). As shown in Figure 1, take $\Phi_p = [0,1] \times [0,1]$ and the S/D binary string $b = 100101$ as an example (in this case, $l = 6$, $n = 2$, and the genotypic precisions $h_1(l) = h_2(l) = \frac{1}{8}$). Let us look how to identify the S/D binary string b and see what each bit value of b means. In Figure 1-(a), the phenotypic domain Φ_p is bisected into four $\Phi_p^{\frac{1}{2}}$ (i.e., the subregions with uniform size $\frac{1}{2}$). According to the *left*-0 and *right*-1 correspondence rule in each coordinate direction, these four $\Phi_p^{\frac{1}{2}}$ then can be identified with $(00), (01), (10)$ and (11). As the phenotype x lies in the

subregion (10) (the gray square), its first building block (BB) should be $BB_1 = 10$. This leads to the first two bits of the S/D binary string b. Likewise, in Figure 1-(b), Φ_p is partitioned into $2^{2\times 2}$ $\Phi_p^{\frac{1}{4}}$, which are obtained through further bisecting each $\Phi_p^{\frac{1}{2}}$ along each direction. Particularly this further divides $\Phi_p^{\frac{1}{2}} = (BB_1)$ into four $\Phi_p^{\frac{1}{4}}$ that can be respectively labelled by $(BB_1, 00), (BB_1, 01), (BB_1, 10)$ and $(BB_1, 11)$. The phenotype x is in $(BB_1, 01)$-subregion (the dark square), so its second BB should be $BB_2 = 01$ and the first four positions of its corresponding S/D binary string b is 1001.

In the same way, Φ_p is partitioned into $2^{2\times 3}$ $\Phi_p^{\frac{1}{8}}$ as shown in Figure 1-(c), with $\Phi_p^{\frac{1}{4}} = (BB_1, BB_2)$ particularly partitioned into four $\Phi_p^{\frac{1}{8}}$ labelled by $(BB_1, BB_2, 00)$, $(BB_1, BB_2, 01)$, $(BB_1, BB_2, 10)$ and $(BB_1, BB_2, 11)$. The phenotype x is found to be $(BB_1, BB_2, 01)$, that is, identical with S/D binary string b. This shows that for any three region partitions, $b = (b_1, b_2, b_3, b_4, b_5, b_6)$, each bit value b_i can be interpreted geometrically as follows: $b_1 = 0$ $(b_2 = 0)$ means the phenotype x is in the left half along the x-coordinate direction (the y-coordinate direction) in Φ_p partition with $\frac{1}{2}$-precision, and $b_1 = 1$ $(b_2 = 1)$ means x is in the right half. Therefore, the first $BB_1 = (b_1, b_2)$ determine the $\frac{1}{2}$-precision location of x. If $b_3 = 0$ $(b_4 = 0)$, it then further indicates that when $\Phi_p^{\frac{1}{2}}$ is refined into $\Phi_p^{\frac{1}{4}}$, the x lies in the left half of $\Phi_p^{\frac{1}{2}}$ in the x-direction (y-direction), and it lies in the right half if $b_3 = 1$ $(b_4 = 1)$. Thus a more accurate geometric location (i.e., the $\frac{1}{4}$-precision location) and a more refined BB_2 of x is obtained. Similarly we can explain b_5 and b_6 and identify BB_3, which determine the $\frac{1}{8}$-precision location of x. This interpretation holds for any high-resolution l bits S/D binary encoding.

3.2 A new genotypic distance on the splicing/decomposable binary representation

For measuring the similarity of the binary strings, the Hamming distance (Hamming, 1980) is widely used on the binary space. Hamming distance describes how many bits are different in two binary strings, but cannot consider the scaled property in non-uniformly binary representations. Thus, the distance distortion between the genotypic and the phenotypic spaces make phenotypically easy problem more difficult. Therefore, to make sure that GAs are able to reliably solve easy problems and problems of bounded complexity, the use of equivalent distances is recommended. For this purpose, we have defined a new genotypic distance on the S/D binary space to measure the similarity of the S/D binary strings.

Definition 1: Suppose any binary strings a and b belong to the S/D binary space Φ_g, the genotypic distance $\|a - b\|_g$ is defined as

$$\|a - b\|_g = \sum_{i=1}^{n} \left| \sum_{j=0}^{l/n-1} \frac{a_{j\times n+i} - b_{j\times n+i}}{2^{j+1}} \right|,$$

where l and n denote the length of the S/D binary strings and the dimensions of the real-encoding phenotypic space Φ_p respectively.

0101 (0.75)	0111 (1.0)	1101 (1.25)	1111 (1.5)
0100 (0.5)	0110 (0.75)	1100 (1.0)	1110 (1.25)
0001 (0.25)	0011 (0.5)	1001 (0.75)	1011 (1.0)
0000 (0.0)	0010 (0.25)	1000 (0.5)	1010 (0.75)

genotypic distances

0101 (0.75)	0111 (0.79)	1101 (0.9)	1111 (1.1)
0100 (0.5)	0110 (0.56)	1100 (0.71)	1110 (0.9)
0001 (0.25)	0011 (0.35)	1001 (0.56)	1011 (0.79)
0000 (0.0)	0010 (0.25)	1000 (0.5)	1010 (0.75)

phenotypic distances

Fig. 2. The genotypic and phenotypic distances between $**\,**$ and 0000 in the S/D binary representation.

For any two S/D binary strings $a, b \in \Phi_g$, we can define the Euclidean distance of their correspond phenotypes:

$$\|a-b\|_p = \sqrt{\sum_{i=1}^{n}(\sum_{j=0}^{l/n-1}\frac{a_{j\times n+i}}{2^{j+1}} - \sum_{j=0}^{l/n-1}\frac{b_{j\times n+i}}{2^{j+1}})^2},$$

as the phenotypic distance between the S/D binary strings a and b. The phenotypic distance $\|\cdot\|_p$ and the genotypic distance $\|\cdot\|_g$ are equivalents in the S/D binary space Φ_g when we consider the convergence process of GAs. We state this as the following theorem.

Theorem 1: The phenotypic distance $\|\cdot\|_p$ and the genotypic distance $\|\cdot\|_g$ are equivalents in the S/D binary space Φ_g because the inequation:

$$\|\cdot\|_p \le \|\cdot\|_g \le \sqrt{n}\times\|\cdot\|_p$$

is satisfied in the the S/D binary space Φ_g, where n is the dimensions of the real-encoding phenotypic space Φ_p.

Proof : For $\forall a, b \in \Phi_g$:

$$\|a-b\|_g = \sum_{i=1}^{n}|\sum_{j=0}^{l/n-1}\frac{a_{j\times n+i}-b_{j\times n+i}}{2^{j+1}}|$$

$$= \sqrt{(\sum_{i=1}^{n}|\sum_{j=0}^{l/n-1}\frac{a_{j\times n+i}-b_{j\times n+i}}{2^{j+1}}|)^2}$$

$$= \sqrt{\begin{array}{l}\sum_{i=1}^{n}(\sum_{j=0}^{l/n-1}\frac{a_{j\times n+i}-b_{j\times n+i}}{2^{j+1}})^2 \\ +\sum_{i_1\neq i_2}^{1\le i_1,i_2\le n}(2\times|\sum_{j=0}^{l/n-1}\frac{a_{j\times n+i}-b_{j\times n+i_1}}{2^{j+1}}| \\ \times|\sum_{j=0}^{l/n-1}\frac{a_{j\times n+i}-b_{j\times n+i_2}}{2^{j+1}}|)\end{array}}$$

because

$$\begin{aligned} 0 \leq & \sum_{i_1 \neq i_2}^{1 \leq i_1, i_2 \leq n} (2 \times |\sum_{j=0}^{l/n-1} \frac{a_{j\times n+i} - b_{j\times n+i_1}}{2^{j+1}}| \\ & \times |\sum_{j=0}^{l/n-1} \frac{a_{j\times n+i} - b_{j\times n+i_2}}{2^{j+1}}|) \\ \leq & (n-1) \sum_{i=1}^{n} (\sum_{j=0}^{l/n-1} \frac{a_{j\times n+i} - b_{j\times n+i}}{2^{j+1}})^2, \end{aligned}$$

then

$$\|a-b\|_p \leq \|a-b\|_g \leq \sqrt{n} \times \|a-b\|_p.$$

Figure 2 shows the comparison of the genotypic distance $\|\cdot\|_g$ and phenotypic distance $\|\cdot\|_p$ between S/D binary strings and 0000 in 2 dimensional phenotypic space, where the length of the S/D binary string $l = 4$. For any two S/D binary strings a and b, if $\|a-0\|_p > \|b-0\|_p$, then $\|a-0\|_g > \|b-0\|_g$ is also satisfied. This means that $\|\cdot\|_p$ and $\|\cdot\|_g$ are equivalent for considering the points' sequence converge to 0. The searching process of GAs can be recognized to explore the points' sequence, which sequentially converge to optimum of the problem. So we can use the new genotypic distance to measure the similarity and convergence of the individuals on the S/D binary place.

The other advantage of the new genotypic distance $\|\cdot\|_g$ is that its computational complexity is $O(l)$ and much lower than the computational complexity $O(l^2)$ of the phenotypic distance $\|\cdot\|_p$. So using the new genotypic distance $\|\cdot\|_g$ can guarantee GA to reliably and predictably solve problems of bounded complexity and improve their performance when consider the similarity of the individuals.

3.3 Theoretical analysis of the splicing/decomposable binary encoding

The above interpretation reveals an important fact that in the new genetic representation the significance of the BB contribution to fitness of a whole S/D binary string varies as its position goes from front to back, and, in particular, the more in front the BB position lies, the more significantly it contributes to the fitness of the whole S/D binary string. We refer such delicate feature of the new representation to as the *BB-significance-variable property*. Actually, it is seen from the above interpretation that the first n bits of an encoding are responsible for the location of the n dimensional phenotype x in a global way (particularly, with $O(\frac{1}{2})$-precision); the next group of n bits is responsible for the location of phenotype x in a less global (might be called 'local') way, with $O(\frac{1}{4})$-precision, and so forth; the last group of n-bits then locates phenotype x in an extremely local (might be called 'microcosmic') way (particularly, with $O(\frac{1}{2^{\ell/n}})$-precision). Thus, we have seen that as the encoding length l increases, the representation

$$\begin{aligned} & (b_1, b_2, \cdots, b_n, b_{n+1}, b_{n+2}, \cdots, b_{2n}, \cdots, \\ & \quad b_{(\ell-n)}, b_{(\ell-n+1)}, \cdots, b_l) \\ = & (BB_1, BB_2, \cdots, BB_{l/n}) \end{aligned}$$

can provide a successive refinement (from global, to local, and to microcosmic), and more and more accurate representation of the problem variables.

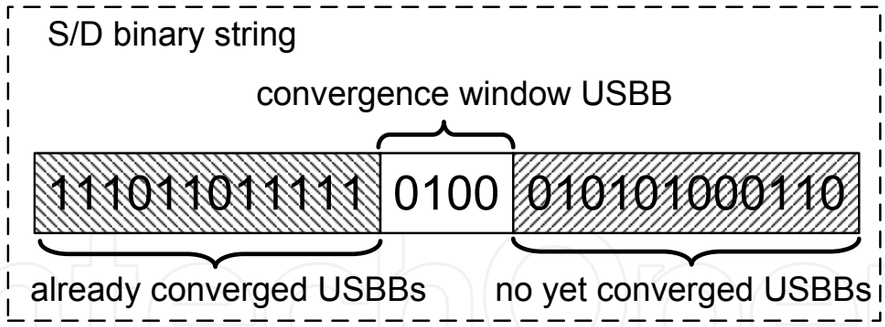


Fig. 3. Domino genotypic at the S/D encodings.

In each BB_i of the S/D binary string, which consists of the bits $(b_{i\times n+1}, b_{i\times n+2}, \cdots, b_{(i+1)\times n})$, $i = 0, \cdots, l/n-1$, these bits are uniformly scaled and independent each other. We refer such delicate feature of BB_i to as the uniform-salient BB (USBB). Furthermore, the splicing different number of USBBs can describe the potential solutions of the problem with different precisions. So, the intra-BB difficulty (within building block) and inter-BB difficulty (between building blocks) (Goldberg, 2002) of USBB are low. The theoretical analysis reveals that GAs searching on USBB can explore the high-quality bits faster than GAs on non-uniformly scaled BB.

The S/D binary encoding is redundancy-free representation because using the S/D binary strings to represent the real values is one-to-one genotype-phenotype mapping. The whole S/D binary string is constructed by a non-uniformly scaled sequence of USBBs. The domino convergence of GAs occurs and USBBs are solved sequentially from high to low scaled.

The BB-significance-variable and uniform-salient BB properties of the S/D binary representation embody many important information useful to the GAs searching. We will explore this information to design new GA based on the S/D binary representation in the subsequent sections.

4. A new S/D binary Genetic Algorithm (SDGA)

The existing exponentially scaled representations including binary and gray encodings consist of non-uniformly scaled BBs. For non-uniformly and competing BBs in the high dimensional phenotype space, there are a lot of noise from the competing BBs lead to a reduction on the performance of GAs. Moreover, by increasing the string length, more and more lower salient BBs are randomly fixed due to the noise from the competing BBs, causing GAs performance to decline. Using large population size can reduce the influence of the noise from the competing BBs. However, in real-world problem, long binary string is necessary to encode a large search space with high precision, and hence we cannot use too large population size to solve the noise problem. Thus, GAs will be premature and cannon converge to the optimum of the problem.

To avoid the noise from the competing BBs of GAs, we have proposed a new splicing/decomposable GA (SDGA) based on the delicate properties of the S/D binary representation. The whole S/D binary string can be decomposed into a non-uniformly scaled sequence of USBBs. Thus, in the searching process of GAs on S/D binary encoding, the

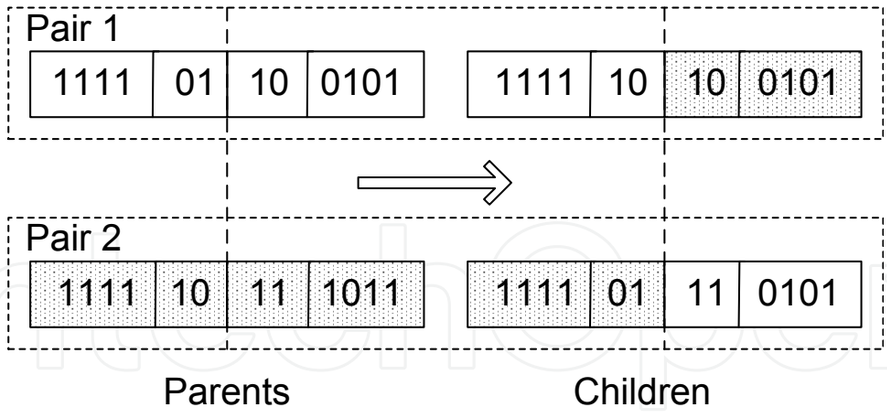


Fig. 4. The genetic crossover and selection in SDGA.

domino convergence occurs and the length of the convergence window is equal to n, the length of USBB. As shown in Figure 3 for 4 dimensional case, the high scaled USBBs are already fully converged while the low scaled USBBs did not start to converge yet, and length of the convergence window is 4.

In the SDGA, genetic operators apply from the high scaled to the low scaled USBBs sequentially. The process of the crossover and selection in SDGA is shown in Figure 4. For two individuals x_1 and x_2 randomly selected from current population, The crossover point randomly set in the convergence window USBB and the crossover operator two children c_1, c_2. The parents x_1, x_2 and their children c_1, c_2 can be divided into two pairs $\{x_1, c_1\}$ and $\{x_2, c_2\}$. In each pair $\{x_i, c_i\}(i = 1, 2)$, the parent and child have the same low scaled USBBs. The select operator will conserve the better one of each pair into next generation according to the fitness calculated by the whole S/D binary string for high accuracy. Thus, the bits contributed to high fitness in the convergence window USBB will be preserved, and the diversity at the low scaled USBBs' side will be maintain. The mutation will operate on the convergence window and not yet converged USBBs according to the mutation probability to increase the diversity in the population. These low salient USBBs will converge due to GAs searching to avoid the noise from the competing BBs. The implementation pseudocode for SDGA algorithm is shown in Figure 5.

Since identifying high-quality bits in the convergence window USBB of GAs is faster than that GAs on the non-uniform BB, while no noise from the competing BBs occurs. Thus, population can efficiently converge to the high-quality BB in the position of the convergence window USBB, which are a component of overrepresented optimum of the problem. According to theoretical results of Thierens (Thierens, 1995), the overall convergence time complexity of the new GA with the S/D binary representation is approximately of order $O(l/\sqrt{n})$, where l is the length of the S/D binary string and n is the dimensions of the problem. This is much faster than working on the binary strings as a whole where GAs have a approximate convergence time of order $O(l)$. The gain is especially significant for high dimension problems.

5. Empirical verification

In this section we present an empirical verification of the performance differences between the different genetic representations and operators we described in the previous sections.

```
Input: N—population size, m—number of USBBs,
       g—number of generations to run;
Termination condition: Population fully converged;
begin
  g ⟵ 0;
  m ⟵ 1;
  Initialize P_g;
  Evaluate P_g;
  while (not termination condition) do
   for t ⟵ 1 to N/2;
   randomly select two individuals x_t^1 and x_t^2 from P_g;
   crossover and selection x_t^1, x_t^2 into P_{g+1};
   end for
   mutation operation P_{g+1};
   Evaluate P_{g+1};
   if (USBB_m fully converged) m ⟵ m + 1;
  end while
end
```

Fig. 5. Pseudocode for SDGA algorithm.

5.1 Two integer benchmark optimization problems

In our experimentation, we use integer-specific variations of the one-max and the fully-deceptive trap problems for a comparison of different genetic representations defined on binary strings.

The integer one-max problem is defined as

$$f_1(x_1, x_2, \cdots, x_n) = \sum_{i=1}^{n} x_i,$$

and the integer deceptive trap is

$$f_2(x_1, x_2, \cdots, x_n) = \begin{cases} \sum_{i=1}^{n} x_i : \text{if each } i,\ x_i = x_{i,max} \\ \sum_{i=1}^{n} x_{i,max} - \sum_{i=1}^{n} x_i - 1 : \text{else.} \end{cases}$$

where $x \in \Phi_p$ and n is the dimension of the problems. In our implementation, we set $n = 30$. For the binary representation, the integer one-max problem is equal to the BinInt problem [Rudnick, 1992]. These two problems have an exponential salience or fitness structure for binary strings. The integer one-max problem is a fully easy problem, whereas the integer deceptive trap should be fully difficult to solve for GAs.

5.2 Comparison of the performance of GAs with different representations

In the first set of experiments we applied a standard GA (SGA) using binary, gray, unary, S/D encodings and SDGA on the integer one-max and deceptive trap problems to compare their performance. We performed 50 runs and each run was stopped after the population was fully converged. That means that all individuals in the population are the same. For fairness of

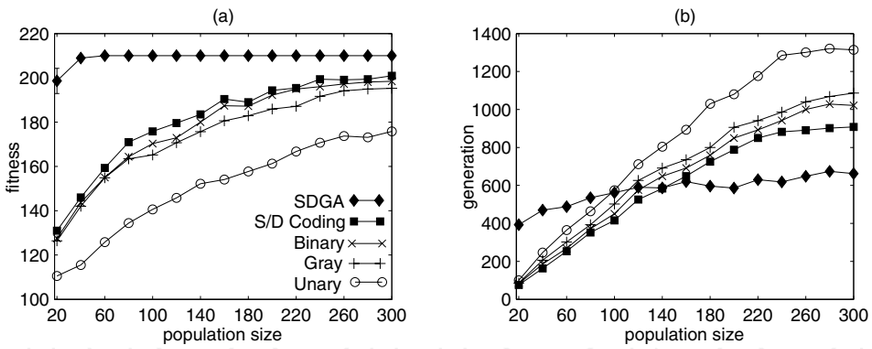


Fig. 6. Integer one-max problem of order 3.

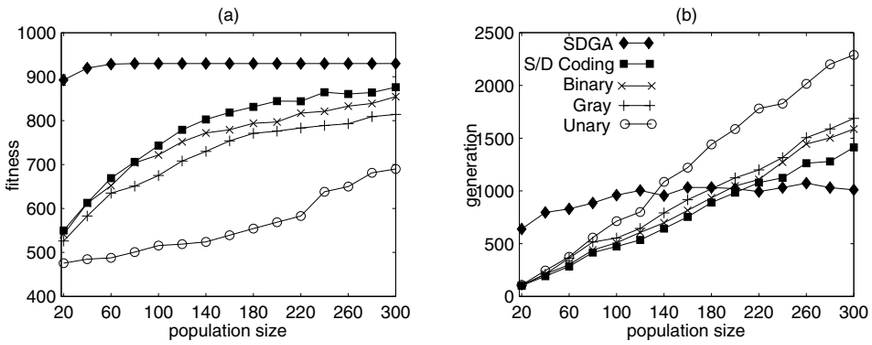


Fig. 7. Integer one-max problem of order 5.

comparison, we implemented SGA with different binary encodings and SDGA with the same parameter setting and the same initial population. For SGA, we used one-point crossover operator (crossover probability=1) and tournament selection operator without replacement of size two. We used no mutation as we wanted to focus on the influence of genetic representations on selectorecombinative GAs.

For the one-max problem, we used 30 dimensional problem for order 2 (in each dimension, the number of different phenotypes $s = 2^2 = 4$), 3 ($s = 2^3 = 8$), 4 ($s = 2^4 = 16$) and 5 ($s = 2^5 = 32$). Because in our implementation, the global optima of deceptive trap problems with low orders cannon be explored by all GAs we used. The deceptive trap problems with high orders are more difficult than those with low orders and are not solvable by GAs. Here, we only present results for the 30 dimensional deceptive trap problems of order 2 ($s = 2^2 = 4$) and 3 ($s = 2^3 = 8$). Using binary, gray and S/D encoding results for the order 2 problems in a string length $l = 60$, for order 3 in $l = 90$, for order 4 in $l = 120$, and for order 5 in $l = 150$. When using unary encoding we need $30 \times 3 = 90$ bits for order 2, $30 \times 7 = 210$ bits for order 3, $30 \times 15 = 450$ bits for order 4 and $30 \times 31 = 930$ bits for order 5 problems.

Figures 6-7 present the results for the integer one-max problem of orders 3 and 5 respectively, and Figures 8-9 show the results for integer deceptive trap problems of orders 2 and 3 respectively. The plots show for SGA with different representations and SDGA the best fitness at the end of the run (left) and the run duration — fully converged generation (right) with respect to the population size N.

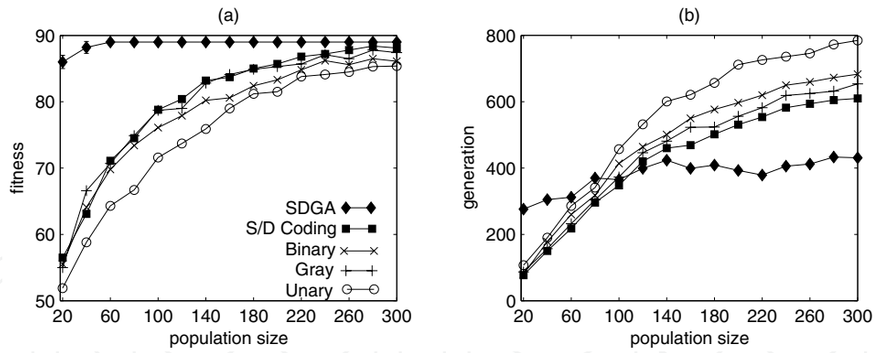


Fig. 8. Deceptive trap problem of order 2.

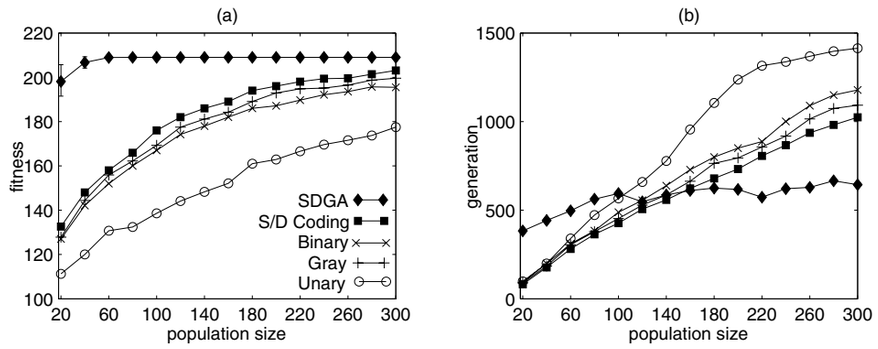


Fig. 9. Deceptive trap problem of order 3.

SGA with different scaled binary representations including binary, gray and S/D encodings complies the noise from the competing BBs. For small population sizes, the noise from the competing BBs strongly occurs and many bits in the binary strings are randomly fixed, so SGA fully converged faster but the best fitness is too bad. That means SGA is premature using small population sizes. For larger population sizes, SGA can explore better solutions, but its run duration is significantly increasing due to the noise from the competing BBs. Furthermore, for these high dimensional problems, the population size increases to 300 still not enough to avoid the noise from the competing BBs, so SGA cannot converge to the optima of the problems, which are overrepresented by BBs.

Due to the problems of the unary encoding with redundancy, which result in an underrepresentation of the optimal solution, SGA using unary encoding perform increasingly badly with increasing problem orders. Therefore, for one-max and deceptive trap problems of order more than three the performance of SGA using unary encoding performance is significantly worse than when using binary, gray and S/D encodings. SGA with gray encoding performs worse than the binary encoding for the one-max problems, and better for the deceptive trap problems.

As expected, SGA using S/D encoding performs better than that using binary and gray encodings for the one-max and the deceptive trap problems. Because in S/D encoding, more salient bits are continuous to construct short and high fit BBs, which are easily identified by SGA. This reveals that the S/D encoding is proper for GAs searching. However, lower salient bits in S/D binary string are randomly fixed by the noise from the competing BBs, the

performance of SGA with S/D encoding cannot significantly better than those with binary and gray encodings.

As shown Figure 6-9, the performance of SDGA is significantly better than SGA with different encodings. Using small population size, the explored solutions when SDGA fully converged are much better than those of SGA because each bit is identified by the searching process of SDGA, and not randomly fixed by the noise from the competing BBs. According to the same reason, the run duration of SDGA is longer than that of SGA. That means there no premature and drift occur. For larger population sizes, the performance of SDGA is much better than that of SGA due to the high-quality solutions and short run duration, because GAs search on USBBs of S/D binary encoding faster than the non-uniformly scaled BBs and domino converge, which occurs only on the non-uniformly sequence of USBBs, is too weak.

P_m	one-max (order 2)		one-max (order 3)		one-max (order 4)	
	best fit. (St. Dev.)	run dur. (St. Dev.)	best fit. (St. Dev.)	run dur. (St. Dev.)	best fit. (St. Dev.)	run dur. (St. Dev.)
SDGA	89.6 (1.24)	383.1 (43.6)	209.2 (2.9)	577.3 (77.4)	448.1 (6.8)	768.7 (107.2)
S/D coding	81.1 (9.8)	446.1 (187.4)	180.9 (21.16)	597 (287)	375.9 (54.3)	694.9 (377.2)
Binary	80.1 (10.3)	473.7 (192.7)	177.7 (21.9)	651 (316.8)	370.5 (42.2)	748.8 (398)
Gray	78.3 (9.6)	496.9 (196.3)	173.1 (20.5)	691.2 (328.5)	365.2 (42.2)	803.6 (434.8)
Unary	76.1 (10.6)	536.8 (218.5)	150.5 (21.3)	844.2 (416.7)	281.5 (26.6)	1006 (558.4)
	one-max (order 5)		decep. (order 2)		decep. (order 3)	
SDGA	926.6 (9.8)	952.9 (118.2)	88.74 (0.78)	380 (48)	208.1 (2.8)	573.1 (75.6)
S/D coding	777.1 (101)	761.8 (422.4)	80.02 (9.7)	428 (173)	182.9 (21.6)	602.9 (285.4)
Binary	752.6 (91)	838.6 (481.6)	77.16 (9.1)	482 (192)	172.8 (21.1)	690.1 (334.8)
Gray	719.8 (87.9)	909.5 (502)	78.76 (9.4)	453 (183)	177.9 (21.8)	647 (309.5)
Unary	560.8 (72.4)	1216 (726.9)	74.18 (10.5)	549 (221)	150.7 (20.6)	882.7 (451.9)

Table 1. Comparison of results of SGA with different binary representations and SDGA for the one-max and deceptive problems.

Table 1 summarizes the experimental results for the one-max and the deceptive trip problems. The best fitness (run duration) of each problem is calculated as the average of the fitness (generations) GAs fully converged with different population sizes.

The average fitness of SDGA is much better than that of other SGA. The standard deviations of best fitness and run duration of SDGA for different problems are significantly smaller than other SGA. That reveals the population size is important parameter for SGA searching, but does not the significant parameter for SDGA searching. The run durations of SDGA for

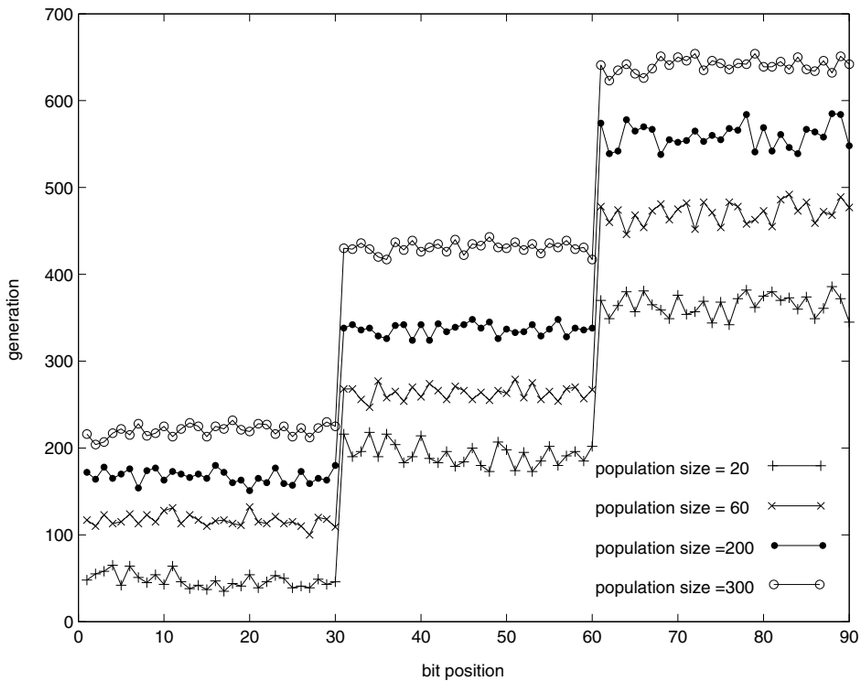


Fig. 10. Convergence process of SDGA without the noise from the competing BBs.

one-max problems with orders 4 and 5 are longer than those of SGA because SGA is strongly premature for the long binary string and small population sizes.

As in Table 1 described, for one-max and deceptive trap problems, all GAs converge to sidewise of the optima, which are overrepresented by BBs. But SGA with different binary representation cannot explore the optima of the problems. The ability of SDGA to explore optima, which are overrepresented by BBs, is significantly better than SGA. To explore the global optimum of the deceptive trap problems, we need use other niche methods to divide the whole population into some sub-populations. In each subpopulation, the global optimum is overrepresented by BBs, thus SDGA can efficiently explore this global optimum of the deceptive trap problems.

5.3 Avoid the noise from the competing BBs

To validate the predictions about avoiding the noise from the competing BBs, We have implemented our SDGA to solve 30 dimensional integer one-max problem of order 3. We have counted the number of generations it takes before each of bits fully converges. Results are averaged over 50 independent runs. Figure 10 shows the bits convergence for a string of length $l = 90$, and the population sizes are $20, 100, 200, 300$ respectively. The experimental results are summered in Table 2. The run duration of each USBB$_i$, $(i = 1, 2, 3)$ is an average of the fully converged generations of the bits, which belong to the USBB$_i$.

As shown in Figure 10 and Table 2, the whole S/D binary string includes three USBBs. In each USBB, the bits converge uniformly at almost same generations. For a non-uniform scaled sequence of USBBs, the domino converge occurs sequentially from high scaled to low scaled

population size	run duration of $USBB_1$	run duration of $USBB_2$	run duration of $USBB_3$
20	47.3(8.2)	193.7(12.7)	365.6(13.8)
100	116.6(6.8)	263.2(7.8)	470.8(12.1)
200	167.4(7.7)	366.5(6.7)	559.6(13.9)
300	220.3(7.0)	430.8(6.6)	633.6(7.8)

Table 2. Comparison of the run durations of USBBs fully converged with different population sizes.(Standard Deviation)

USBBs. Thus, no less salient bit converges with more salient bit at same generations and no noise from the competing BBs occurs.

On the other hand, we know the noise from the competing BBs strongly occurs when GAs using a small population size. In our implementations, when the population size of SDGA is small to 20, the convergence process of bits is as same as SDGA using large population size. The low scaled USBBs converge during long generations by SDGA and no noise from the competing BBs occurs.

It is clear form Figure 10 and Table 2 that the predictions and the experimental results coincide very well.

5.4 SDGA with the mutation operator

In this subsection we have consider the action of the mutation operator for SDGA searching. We have implemented our SDGA with different mutation probabilities to solve 30 dimensional integer one-max problem of order 3. Results are averaged over 50 independent runs. Figure 11 presents the experimental results where mutation probabilities are $0.001, 0.005, 0.01, 0.05$ and 0.1 respectively. The plots show for SDGA the run duration — fully converged generations with respect to the population size N.

As shown in Figure 11, when the mutation probabilities are smaller than 0.01, SDGA can fully converge with small and large population sizes and the run durations do not increase too long. When the mutation probabilities increase larger than 0.01, SDGA with large population sizes are difficult to fully converge, and only when using small population sizes, SDGA can fully converge, but the run durations increase significantly.

Table 3 summaries the experimental results with population sizes 20, 40 and 60. For small population sizes (20 and 40), the mutation operators can improve the performance of SDGA, because it can find some high-quality bits, which are not included in current population. For large population sizes ($\geq$ 60), all high-quality bits are included in the initial population, so mutation operator cannot improve the best fitness when SDGA fully converged. Furthermore, when the mutation probability is large than 0.01, SDGA cannot fully converge in a reasonable time (here we set the upper bound of the run duration equal to 10^6 generations).

5.5 Genotypic distance on the S/D binary representation

To validate the predictions about the methods depended on the distance of real-valued space, can be directly used on the S/D binary space based on our new defined genotypic distance, we have combined SGA with the S/D binary encoding and the dynamic niche sharing methods

P_m	$N = 20$		$N = 40$		$N = 60$	
	best fit. (St. Dev.)	run dur. (St. Dev.)	best fit. (St. Dev.)	run dur. (St. Dev.)	best fit. (St. Dev.)	run dur. (St. Dev.)
0	198.6 (5.7)	393 (72)	208.9 (1.2)	470 (55)	210 (0)	488 (54)
0.001	201.7 (100)	411 (49)	209.4 (1.2)	472 (43)	210 (0)	517 (54)
0.005	202.7 (2.9)	422 (55)	208.9 (1.3)	492 (82)	210 (0)	535 (89)
0.01	203.8 (2.2)	415 (59)	209.1 (1.2)	504 (76)	210 (0)	545 (80)
0.05	209.3 (1)	534 (158)	209.9 (0.3)	739 (202)	210 (0)	1202 (317)
0.1	209.8 (0.6)	688 (133)	210 (0)	5629 (1857)	210 (0)	66514 (21328)
0.2	209.8 (0.4)	10981 (7668)	– (–)	– (–)	– (–)	– (–)

Table 3. Comparison of results of SDGA with different mutation probabilities for one-max problem of order 3. ("-": cannot fully converged during 10^6 generations)

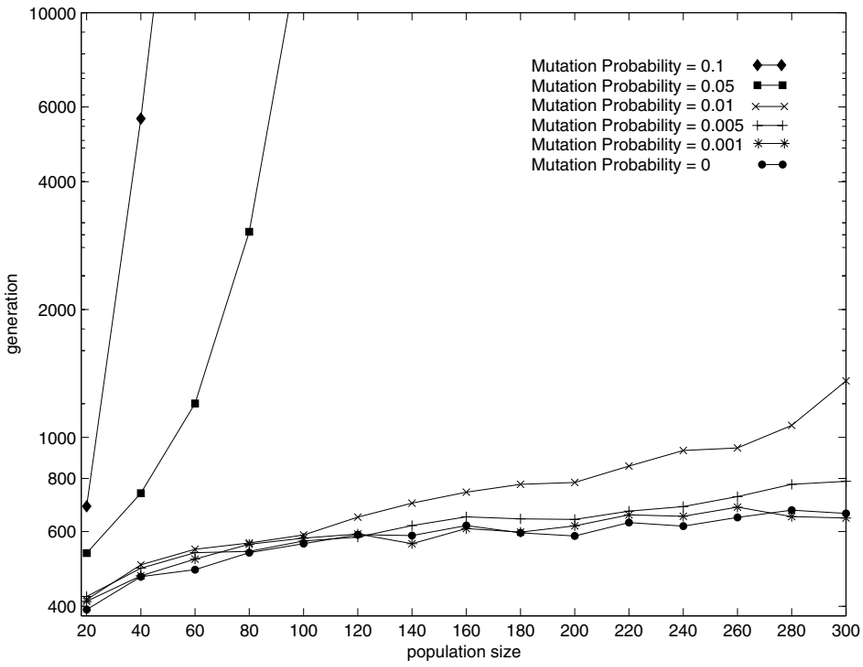


Fig. 11. SDGA with the mutation operator by different mutation probabilities for one-max problem of order 3.

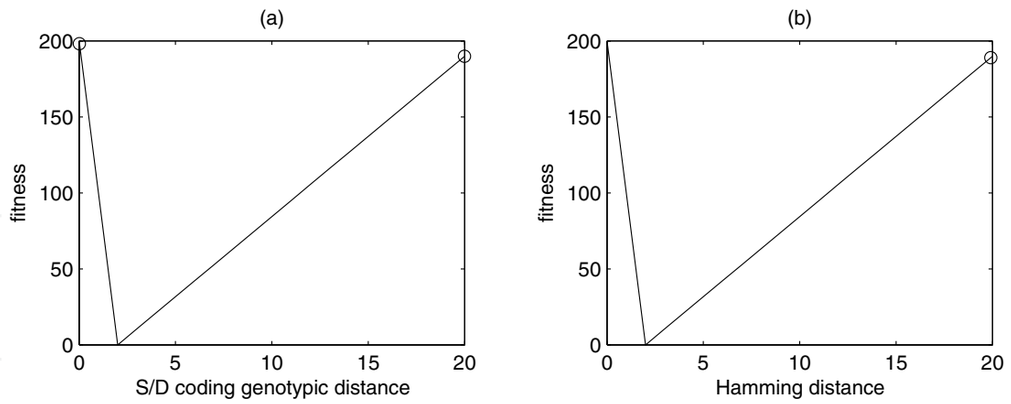


Fig. 12. Comparison of results of the dynamic niche sharing methods with S/D genotypic distance and Hamming distance for $f_3(x)$. (key: "o" — the optima in the final population)

[Miller] for multimodal function optimization to solve 4 benchmark multimodal optimization problems as listed in Table 4. To assess the effectiveness of the new genotypic distance on the S/D binary space, its performance is compared with the combination of SGA with S/D binary representation and the dynamic niche sharing methods based on Hamming distance. In applying SGA, we set the initial population size $N = 100$, the maximal generations $g_{mx} = 1000$, the length of S/D binary string for each dimension $l/n = 32$, the crossover probability $p_c = 0.8$ and the mutation probability $p_m = 0.005$.

Two-peak trap function (2 peaks):

$$f_3(x) = \begin{cases} \frac{200}{2}(2-x), \text{ for } 0 \le x < 2; \\ \\ \frac{190}{18}(x-2), \text{ for } 2 \le x \le 20; \end{cases}$$

Deb's function (5 peaks):

$$f_4(x) = \sin^6(5\pi x), x \in [0,1];$$

Deb's decreasing function (5 peaks):

$$f_5(x) = 2^{-2((x-0.1)/0.9)^2} \sin^6(5\pi x), x \in [0,1];$$

Roots function (6 peaks):

$$f_6(x) = \frac{1}{1+|x^6-1|}, \text{where } x \in C, x = x_1 + ix_2 \in [-2,2];$$

Table 4. The test suite of multimodal functions used in our experiments.

Figures 12 - 15 show the comparison results of the dynamic niche sharing methods with the S/D genotypic distance and Hamming distance for $f_3(x) - f_6(x)$, respectively. Table 5 lists the solution quality comparison results in terms of the numbers of multiple optima maintained. We have run each algorithm 10 times. The dynamic niche sharing methods with the S/D

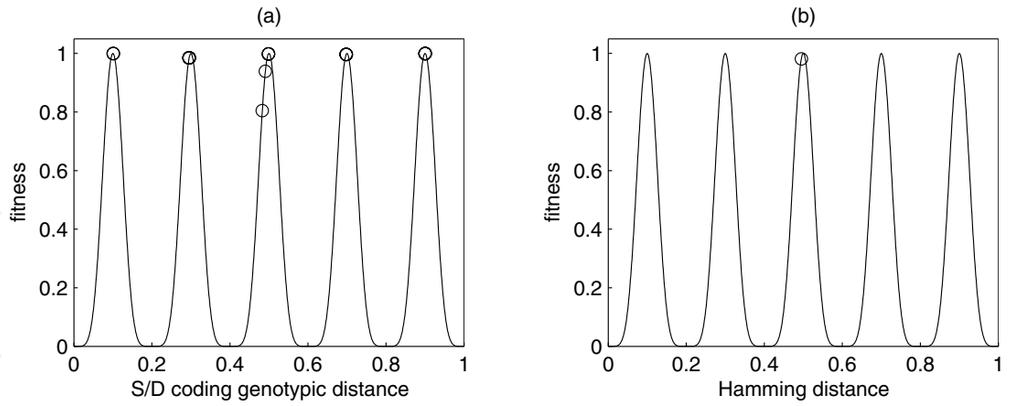


Fig. 13. Comparison of results of the dynamic niche sharing methods with S/D genotypic distance and Hamming distance for $f_4(x)$. (key: "o" — the optima in the final population)

genotypic distance can explore all optima in $f_3(x) - f_6(x)$ at each run. Contrary, for the niche methods with Hamming distance, the final population converged to a single optimum of the multimodal problem and cannot find multiply optima. That means the niche method cannon work due to the distance distortion between genotypic space (S/D binary space) and phenotypic space (real-valued space) when using Hamming distance.

The experimental investigations reveal that the methods depended on the Euclidean distance on the real-valued space can be directly used on the S/D binary space with our new defined genotypic distance.

	Distance	S/D genotypic distance		Hamming distance	
	threshold	Optima No.	Success rate	Optima No.	Success rate
f_3	2.0	2	100%	1	0%
f_4	0.16	5	100%	1	0%
f_5	0.16	5	100%	1	0%
f_6	0.8	6	100%	1	0%

Table 5. Comparison of results of the dynamic niche sharing methods with the S/D genotypic distance and Hamming distance.

6. Discussion

This paper has given for the first time a uniform-salient building block (USBB) in the S/D binary representation, which include uniformly scaled bits. This assumes that the phenotypic space Φ_p is uniformly scaled in each dimension. If the assumption is not be satisfied, we need to normalize the phenotypic space Φ_p first, then encoding the normalized phenotypic space Φ'_p into the S/D binary space Φ_g to guarantee that the bits in each USBB have same scaled.

SDGA applies on the S/D binary representation and converges from high scaled to low scaled USBBs sequentially. However, when the convergence window USBB cannon converge to single high-quality BB, there maybe are some high-quality BBs existing to describe different optima of the problem. At this time, we need to use some other methods (e.g. the niche methods) to divide the whole population into several sub-populations and each sub-population focus on each optimum. Thus, each optimum will be overrepresented by

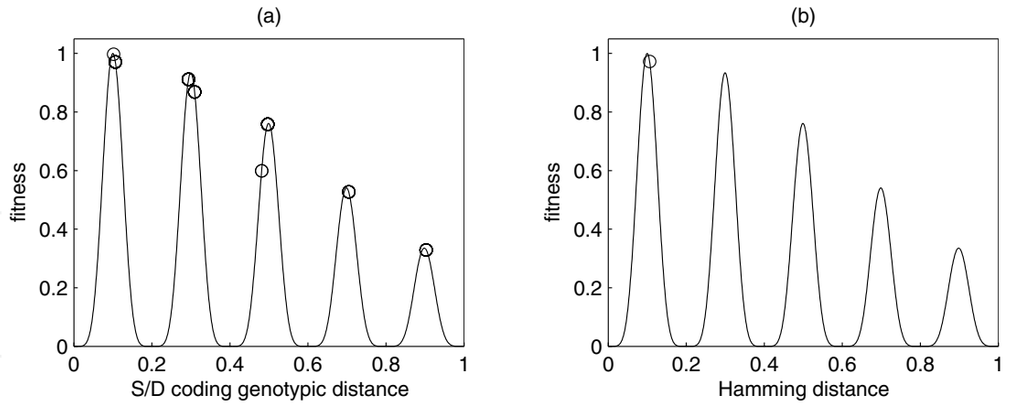


Fig. 14. Comparison of results of the dynamic niche sharing methods with S/D genotypic distance and Hamming distance for $f_5(x)$. (key: "o" — the optima in the final population)

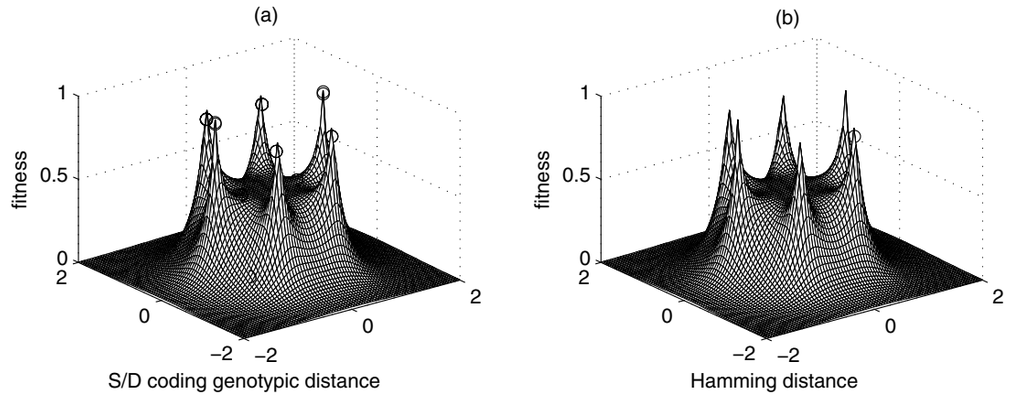


Fig. 15. Comparison of results of the dynamic niche sharing methods with S/D genotypic distance and Hamming distance for $f_6(x)$. (key: "o" — the optima in the final population)

BBs in its sub-population and SDGA can efficiently explore all the optima using these sub-populations.

7. Conclusions

In this paper, we introduce a new genetic representation — a splicing/decomposable (S/D) binary encoding, which was proposed based on some theoretical guidance and existing recommendations for designing efficient genetic representations. The S/D binary representation can be spliced and decomposed to describe potential solutions of the problem with different precisions by different number of uniform-salient building blocks (USBBs). According to the characteristics of the S/D binary representation, genetic and evolutionary algorithms (GEAs) can be applied from the high scaled to the low scaled BBs sequentially to avoid the noise from the competing BBs and improve GEAs' performance. Our theoretical and empirical investigations reveal that the S/D binary representation is more proper than other existing binary encodings for GEAs searching. Moreover, we define a new genotypic distance on the S/D binary space, which is equivalent to the Euclidean distance on the real-valued space during GEAs convergence. Based on the new genotypic distance, GEAs can reliably

and predictably solve problems of bounded complexity and the methods depended on the Euclidean distance for solving different kinds of optimization problems can be directly used on the S/D binary space.

8. Acknowledgment

This research was supported by Macau Science and Technology Develop Funds (Grant No. 021/2008/A) and (Grant No. 017/2010/A2) of Macau Special Administrative Region of the People's Republic of China.

9. References

Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning. Reading, MA: Addison-Wesley.

Hamming, R. (1980). Coding and information theory. Prentice-Hall. Han, K. H. & Kim, J. H. (2000). Genetic quantum algorithm and its application to combinatorial optimization problem, Proceeding of Congress on Evolutionary Computation 2000: Volume 1, pp. 1354-1360,

La Jolla, CA. Harik, G. R., Cantu-Paz, E., Goldberg, D. E. & Miller, B. L. (1997). The gambler's ruin problem, genetic algorithms and the size of populations. In Back, T. (Ed.), Proceedings of the Forth International Conference on Evolutionary Computation, pp. 7-12, New York.

Holland, J. H. (1975). Adaptation in natural and artificial systems. Ann Arbor, MI: University of Michigan Press. Julstrom, B. A. (1999). Redundant genetic encodings may not be harmful. Proceedings of the Genetic and Evolutionary Computation Conference 1999: Volume 1. San Francisco, CA: Morgan Kaufmann Publishers.

Leung, K. S., Sun, J. Y. & Xu, Z. B. (2002). Efficiency speed-up strategies for evolutionary computation: an adaptive implementation. Engineering Computations, 19 (3), pp. 272-304.

Leung, K. S. & Liang, Y. (2003). Adaptive elitist-population based genetic algorithm for multimodal function optimization. Proceeding of Genetic and Evolutionary Computation Conference 2003: Volume 1, pp. 1160-1171, Chicago, USA.

Liang, Y. & Leung, K. S. (2006). Evolution Strategies with Exclusion-based Selection Operators and a Fourier Series Auxiliary Function, Applied Mathematics and Computation, Volume 174, pp. 1080-1109.

Liepins, G. E. & Vose, M. D. (1990). Representational issues in genetic optimization. Journal of Experimental and Theoretical Artificial Intelligence, 2, pp.101-115.

Lobo, F. G., Goldberg. D. E. & Pelikan, M. (2000). Time complexity of genetic algorithms on exponentially scaled problems. Proceedings of the Genetic and Evolutionary Computation Conference 2000: Volume 1. San Francisco, CA: Morgan Kaufmann Publishers.

Mahfoud, S. W. (1996). Niching methods for genetic algorithms. Doctoral Thesis, University of Illinois at Urbana-Champaign.

Miller, B. L. & Goldberg, D. E. (1996). Optimal sampling for genetic algorithms (IlliGAL Report No. 96005). Urbana, IL: University of Illinois at Urbana-Champaign.

Rothlauf, F. (2002). Representations for genetic and evolutionary algorithms. Heidelberg; New York: Physica-Verl., 2002 Schaffer, J. D. (Ed.) (1989a). Proceedings of the

Third International Conference on Genetic Algorithms. San Francisco, CA: Morgan Kaufmann Publishers

Schaffer, J. D., Caruana, R. A., Eshelman, L. J. & Das, R. (1989b). A study of control parameters affecting online performance of genetic algorithms for function optimization. Proceedings of the Third International Conference on Genetic Algorithms. San Mateo, CA: Morgan Kaufmann.

Sendhoff, B., Kreutz, M. & von Seelen, W. (1997). A condition for the genotype-phenotype mapping: Causality. In Back, T. (ed.), Proceedings of the Seventh International Conference on Genetic Algorithms, pp. 73-80, San Francisco: Morgan Kaufmann.

Thierens, D. (1995). Analysis and design of genetic algorithms. Leuven, Belgium: Katholieke Universiteit Leuven.

Whitley, D. (2000). Local search and high precision gray codes: Convergence results and neighborhoods. In Martin, W., & Spears, W. (Eds.), Foundations of Genetic Algorithms 6. San Francisco, California: Morgan Kaufmann Publishers, Inc.

Wong, Y. Y., Lee, K. H., Leung, K. S. & C.W. Ho, C. W. (2003). A novel approach in parameter adaptation and diversity maintenance for genetic algorithm. Soft Computing, 7(8), pp. 506-515.

Wong, Z. Y., Leung, K. S., Wong, M. L. & Fang, J. (2000). A new type of nonlinear integrals and the computational algorithm. Fuzzy Sets and System, 112, pp. 223-231.

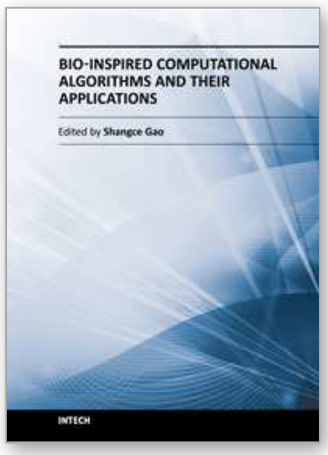
Xu, K. B., Wang, Z. Y., Heng, P. A. & Leung, K. S. (2003a). Classification by Nonlinear Integral Projections. IEEE Transactions on Fuzzy Systems, 11(2), pp. 187 - 201.

Xu, Z. B., Leung, K. S., Liang, Y. & Leung, Y. (2003b). Efficiency speed-up strategies for evolutionary computation: fundamentals and fast-GAs. Applied Mathematics and Computation 142, pp. 341-388.

Zhu, Z. Y. & Leung, K. S. (2002a). An enhanced annealing genetic algorithm for multi-objective optimization problems, Proceeding of Genetic and Evolutionary Computation Conference 2002, New York, USA.

Zhu, Z. Y. & Leung, K. S. (2002b). Asynchronous self-adjustable island genetic algorithm for multi-objective optimization problems, Proceeding of Congress on Evolutionary Computation 2002, Hawaii, USA.

Liang, Y. & Leung, K. S. (2011). Genetic Algorithm with Adaptive Elitist-population Strategies for Multimodal Function Optimization. Applied Soft Computing, 11(2), pp. 1160-1171.



Bio-Inspired Computational Algorithms and Their Applications
Edited by Dr. Shangce Gao

ISBN 978-953-51-0214-4
Hard cover, 420 pages
Publisher InTech
Published online 07, March, 2012
Published in print edition March, 2012

Bio-inspired computational algorithms are always hot research topics in artificial intelligence communities. Biology is a bewildering source of inspiration for the design of intelligent artifacts that are capable of efficient and autonomous operation in unknown and changing environments. It is difficult to resist the fascination of creating artifacts that display elements of lifelike intelligence, thus needing techniques for control, optimization, prediction, security, design, and so on. Bio-Inspired Computational Algorithms and Their Applications is a compendium that addresses this need. It integrates contrasting techniques of genetic algorithms, artificial immune systems, particle swarm optimization, and hybrid models to solve many real-world problems. The works presented in this book give insights into the creation of innovative improvements over algorithm performance, potential applications on various practical tasks, and combination of different techniques. The book provides a reference to researchers, practitioners, and students in both artificial intelligence and engineering communities, forming a foundation for the development of the field.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen