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

Yong Liang
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., HK, China
yliang@cse.cuhk.edu.hk

Kwong-Sak Lueng
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., HK, China
ksleung@cse.cuhk.edu.hk

Kin-Hong Lee
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., HK, China
khlee@cse.cuhk.edu.hk

ABSTRACT

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. Our theoretical and empirical investigations reveal that the S/D binary representation is more proper than other existing binary encodings for searching of genetic algorithms (GAs). 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 GAs convergence. Based on the new genotypic distance, GAs 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.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*heuristic methods.*

General Terms

Algorithms

Keywords

Genetic algorithm, genetic representation

1. INTRODUCTION

Most of the real-world problems could be encoded by different representations, but genetic algorithms (GAs) 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 GAs on the real-world problems [1] [7] [12].

A large number of theoretical and empirical investigations on genetic representations were made over the last decades, and have shown that the behavior and performance of GAs is strongly influenced by the representation used. Originally, the schema theorem and the building block hypothesis proposed by [1] and [4] to model the performance of GAs to process similarities between binary bitstrings. 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 [9] studied the performance differences of GAs by different binary representations for real encoding.

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 problems [5], 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 [9].

The binary encoding uses exponentially scaled bits to represent phenotypes. Its genotype-phenotype mapping is a one-to-one mapping and encodes phenotypes without redundancy. 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. In addition, the binary encoding has the effect that genotypes of some phenotypical neighbors are completely different. As a result, the locality of the binary representation is partially low, i.e. *Hamming cliff* [10]. 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.

The non-redundant gray encoding [10] was designed to overcome the problems with the *Hamming cliff* of the binary encoding. 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 [3] [12].

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 [1] has proposed two basic design principles for encodings: (i), Principle of minimal alphabets — the alphabet of the encoding should be as small as possible while still allowing a natural representation of solutions; and (ii), 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.

This paper is organized as follows. Section 2 introduces a novel splicing/decomposable (S/D) binary representation and its genotypic distance. Section 3 proposes the new genetic algorithm based on the S/D binary representation, the splicing/Decompocable genetic algorithm (SDGA). Section 4 discusses the performance of the SDGA and compares the S/D binary representation with other existing binary encodings from the empirical studies. The paper conclusion are summarized in Section 5.

2. A NOVEL SPLICING/DECOMPOSABLE BINARY GENETIC REPRESENTATION

Based on above investigation results and recommendations, Leung et al. have proposed a new genetic representation, which is proper for GAs searching [6] [13]. 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).

2.1 A Splicing/Decomposable Binary Encoding

In [6], Leung et al. 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

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

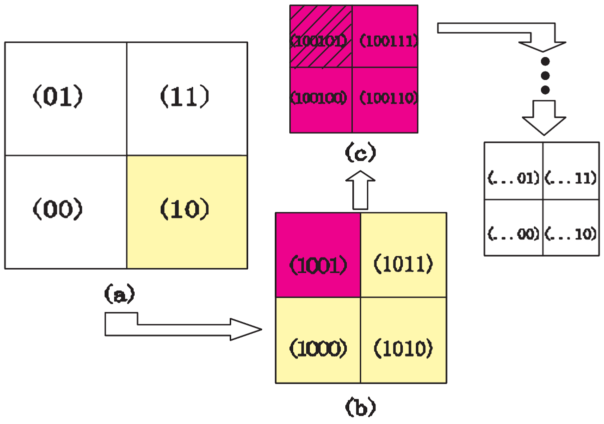

Figure 1: A graphical illustration of the splicing/decomposable representation scheme, where (b) is the refined bisection of the gray cell (10) in (a) (with mesh size $O(1/2)$), (c) is the refined bisection of the dark cell (1001) in (b) (with mesh size $O(1/2^2)$), and so forth.

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

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

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 $2^{2\times2}$ $\Phi_p^{\frac{1}{4}}$, 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.

2.2 A New Genotypic Distance on the Splicing/Decomposable Binary Representation

For measuring the similarity of the binary strings, the Hamming distance [2] 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 makes 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 define 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

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

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.

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

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

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.

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

Figure 2: The genotypic and phenotypic distances between $****$ and 0000 in the S/D binary representation.

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

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

$$\begin{aligned}
\|a-b\|_g &= \sum_{i=1}^{n} \Big| \sum_{j=0}^{l/n-1} \frac{a_{j\times n+i} - b_{j\times n+i}}{2^{j+1}} \Big| \\
&= \sqrt{\Big(\sum_{i=1}^{n} \Big| \sum_{j=0}^{l/n-1} \frac{a_{j\times n+i} - b_{j\times n+i}}{2^{j+1}} \Big|\Big)^2} \\
&= \sqrt{\begin{array}{l} \sum_{i=1}^{n} \big(\sum_{j=0}^{l/n-1} \frac{a_{j\times n+i} - b_{j\times n+i}}{2^{j+1}}\big)^2 \\ + \sum_{i_1 \ne i_2}^{1\le i_1, i_2 \le n} (2 \times \big|\sum_{j=0}^{l/n-1} \frac{a_{j\times n+i} - b_{j\times n+i_1}}{2^{j+1}}\big| \\ \times \big|\sum_{j=0}^{l/n-1} \frac{a_{j\times n+i} - b_{j\times n+i_2}}{2^{j+1}}\big|) \end{array}}
\end{aligned}$$

because

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

then

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

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. According to the distance distortion of the genetic representation, using the new genotypic distance $\|\cdot\|_g$ can guarantee GA to reliably and predictably solve problems of bounded complexity.

2.3 Theoretical Analysis of the Splicing/Decomposable Binary Encoding

First, we present the two integer-specific variations of the one-max and the fully-deceptive trap problems we want to

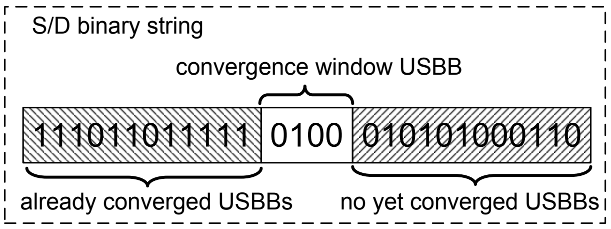

Figure 3: Domino genotypic at the S/D encodings.

use for analyses and comparisons 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.

The interpretation in the previous sections 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.

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. 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 rough approximations of the problem solutions with different precisions. So, the intra-BB difficulty (within building block) and inter-BB difficulty (between building blocks) [1] 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.

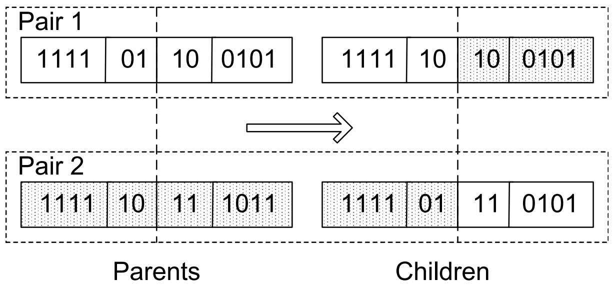

Figure 4: The genetic crossover and selection in SDGA.

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.

3. A NEW S/D BINARY GENETIC ALGORITHM (SDGA)

The above interpretation reveals that for non-uniformly binary strings and competing Building Blocks (BBs) in binary and grid encodings, there are a lot of noise from the competing BBs lead to a reduction on the performance of GAs. To avoid this problem, we propose 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 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 is randomly set in the convergence window USBB and the crossover operator generates 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 selection 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 genetic drift. The implementation outline of the SDGA is shown in Figure 5.

```
Input: $N$—population size, $m$—number of USBBs,
       $g$—number of generations to run;
Termination condition: Population fully converged;

begin
  $g \longleftarrow 0$;
  $m \longleftarrow 1$;
  Initialize $P_g$;
  Evaluate $P_g$;
  while (not termination condition) do
  for $t \longleftarrow 1$ 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) $m \longleftarrow m+1$;
  end while
end
```

Figure 5: Pseudocode for SDGA algorithm.

Since identifying high-quality bits in the convergence window USBB of GAs is faster than that GAs on the non-uniform BB, while no genetic drift 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 [11], 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.

4. 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.

4.1 Comparison of the Performance of GAs with Different Representations

In our experimentation, we use 30 dimensional one-max and deceptive trap problems for a comparison of different genetic representations defined on binary strings. For the binary representation, the integer one-max problem is equal to the BinInt problem [9]. 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.

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 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.

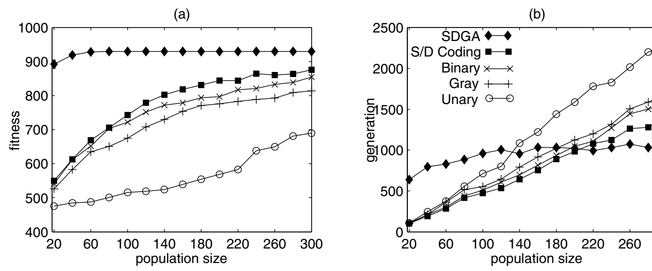

Figure 6: Integer one-max problem of order 5.

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 and 7 present the results for the integer one-max problem of order 5 and the results for integer deceptive trap problem of order 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.

SGA with different scaled binary representations including binary, gray and S/D encodings complies domino convergence, genetic drift and noise from BBs. For small population sizes, genetic drift 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 domino convergence. Furthermore, for these high dimensional problems, the population size increases to 300 still not enough to avoid the noise from 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

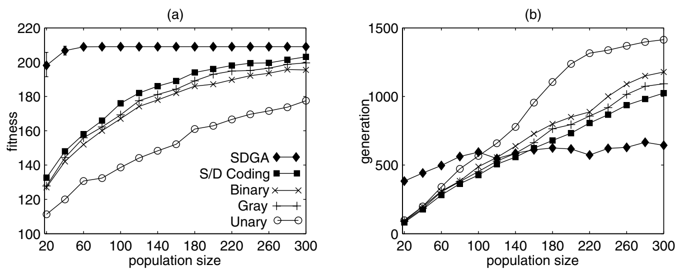

Figure 7: Deceptive trap problem of order 3.

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 genetic drift and noise from BBs, the performance of SGA with S/D encoding cannot significantly better than those with binary and gray encodings.

As shown Figures 6 and 7, 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 genetic drift and noise from 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 convergence, which occurs only on the non-uniformly sequence of USBBs, is too weak.

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 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.

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

P_m	one-max (order 2)		one-max (order 3)		one-max (order 4)	
	best fit. (s. d.)	run dur. (s. d.)	best fit. (s. d.)	run dur. (S. d.)	best fit. (s. d.)	run dur. (s. d.)
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)

4.2 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 8 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 8, 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 2 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

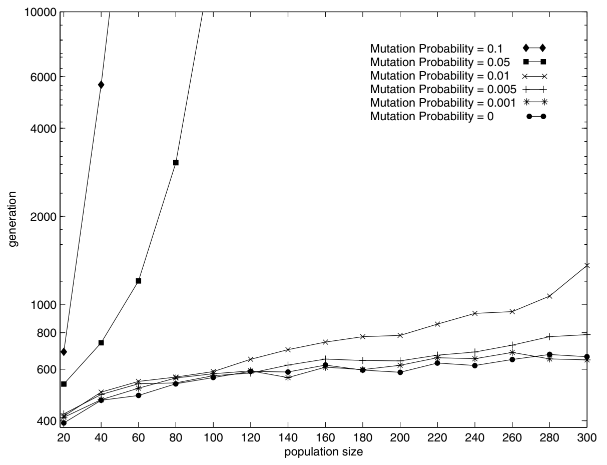

Figure 8: SDGA with the mutation operator by different mutation probabilities for one-max problem of order 3.

sizes (≥ 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).

Table 2: Comparison of results of SDGA with different mutation probabilities for one-max problem of order 3. ("-": cannot fully converged during 10^6 generations)

P_m	$N = 20$		$N = 40$		$N = 60$	
	best fit. (s. d.)	run dur. (s. d.)	best fit. (s. d.)	run dur. (s. d.)	best fit. (s. d.)	run dur. (s. d.)
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)	− (−)	− (−)	− (−)	− (−)

4.3 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 [8] for multimodal function optimization to solve 4 benchmark multimodal optimization problems as listed in Table 3. 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 $g_{max} = 1000$, the length of S/D binary string for each dimension $l/n = 32$, the crossover probability $p_c = 0.8$ and the mutation probability $p_m = 0.005$.

Figure 9 shows the comparison results of the dynamic niche sharing methods with the S/D genotypic distance and Hamming distance for $f_6(x)$. Table 4 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 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.

Table 3: The test suite of multimodal functions used in our experiments.

Two-peak trap function (2 peaks):

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

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):

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

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.

Table 4: Comparison of results of the dynamic niche sharing methods with the S/D genotypic distance and Hamming distance.

	Distance threshold	S/D genotypic distance		Hamming distance	
		Opti. No.	Success rate	Opti. 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%

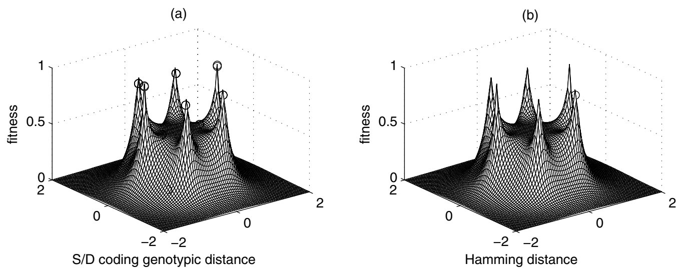

Figure 9: Comparison of results of the dynamic niche sharing methods with S/D genotypic distance and Hamming distance for $f_6(x)$. (key: "o" — the optima in the final population)

5. CONCLUSIONS

In this paper, we introduced 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 the rough approximations of the problem solutions with different precisions by different number of uniform-salient building blocks (USBBs). According to the characteristics of the S/D binary representation, genetic algorithms (GAs) can be applied from the high scaled to the low scaled BBs sequentially to avoid genetic drift and noise of the competing BBs and improve GAs' performance. Our theoretical and empirical investigations reveal that the S/D binary representation is more proper than other existing binary encodings for GAs searching. Moreover, we defined a new genotypic distance on the S/D binary space, which is equivalent to the Euclidean distance on the real-valued space during GAs convergence. Based on the new genotypic distance, GAs 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.

6. ACKNOWLEDGMENTS

This research was partially supported by RGC Earmarked Grant 4173/04E of Hong Kong SAR and RGC Research Grant Direct Allocation of the Chinese University of Hong Kong.

7. REFERENCES

[1] D. E. Goldberg, D. E, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA: Addison-Wesley, 1989

[2] R. Hamming, *Coding and Information Theory*, Prentice-Hall, 1980

[3] K. H. Han, and J. H. Kim, Genetic quantum algorithm and its application to combinatorial optimization problem, *Proceeding of Congress on Evolutionary Computation 2000*, 1: 1354-1360, 2000

[4] J .H. Holland, *Adaptation in Natural and Artificial systems*, Ann Arbor, MI: University of Michigan Press, 1975

[5] B. A. Julstrom, Redundant genetic encodings may not be harmful, *Proceedings of the Genetic and Evolutionary Computation Conference 1999*, 1: 791, San Francisco, CA: Morgan Kaufmann Publishers, 1999

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

[7] G. E. Liepins, and M. D. Vose, Representational issues in genetic optimization, *Journal of Experimental and Theoretical Artificial Intelligence*, 2: 101-115, 1990

[8] S. W. Mahfoud, *Niching methods for genetic algorithms*, Doctoral Thesis, University of Illinois at Urbana-Champaign, 1996

[9] F. Rothlauf, *Representations for Genetic and Evolutionary Algorithms*, Heidelberg; New York: Physica-Verl., 2002

[10] J. D. Schaffer, R. A. Caruana, L. J. Eshelman, and R. Das, 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, 1989

[11] D. Thierens, *Analysis and Design of Genetic Algorithms*, Leuven, Belgium: Katholieke Universiteit Leuven, 1990

[12] D. Whitley, 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., 2000

[13] Z. B. Xu, K. S. Leung, Y. Liang, and Y. Leung, Efficiency speed-up strategies for evolutionary computation: fundamentals and fast-GAs, *Applied Mathematics and Computation*, 142: 341-388, 2003