International Journal of Computational Intelligence Systems, Vol.2, No. 1 (March, 2009), 91-98

On the Supply of Superior Order-1 Building Blocks for a Class of Periodical Fitness Functions

Hongqiang MO
College of Automation Sci. and Eng., South China University of Technology, 510641 Guangzhou, P.R. China, Email: hqiangmo@scut.edu.cn,

Zhong LI
Faculty of Mathematics and Computer Science, FernUniversität in Hagen, 58084 Hagen, Germany Corresponding author, Email: zhong.li@fernuni-hagen.de

JinBae PARK
Dept. of Electrical and Computer Eng., Yonsei University, Seoul, 120-749, Korea

YoungHoon JOO
School of Electronic and Information Eng., Kunsan University, Kunsan, Chunbuk, 573-701, Korea

Received: 18-09-2008 Revised: 29-01-2009

Abstract

In addition to GA-deception, the lack of fitness differences among low-order schemata can also degrade GA's search. Therefore, a coding should present adequate superior low-order building blocks at the early stage of search. This paper aims to reveal the inherent periodicity in the search process of a genetic algorithm, and to show how to make use of this periodicity in the design of representation for fitness functions with periods of the reciprocals of positive integers so as to ensure the effective supply of superior order-1 building blocks. Finally, simulations are given to illustrate the effectiveness of the proposed method.

Keywords - genetic algorithms; periodical function; linear weighted coding; building block; cardinality of coding alphabet

1. Introduction

The capacity of the genetic algorithms (GA's) to simultaneously process a large number of schemata leads to their success in exploring complex fitness. [1, 2] In order to make use of this parallel search capacity, coding and population size should be properly chosen so as to provide adequate initial building blocks for search.[1-5]

However, an adequate supply of raw building blocks itself does not ensure the formation and growth of superior building blocks, and the effects of building blocks' fitness to the performance of GA's search should also be taken into account. Some features of fitness landscapes are relevant to GA's performance, and determine how schemata interact and combine during the typical evolution of the GA's search.[6] Moreover, representation can also affect building block dynamics and GA's ability to balance exploration and exploitation of building blocks.[7]

Walsh analysis has been used to analyze the effects of coding and genetic operators to GA's behaviors. [2, 8-12] The analyses on schema fitness value reveal that GA-hardness and GA-deception might mislead the search [8-11], and non-inferior building blocks might cause a GA to get stuck into a local optimum because of a synchronization problem.[12] However, there are still some kinds of fitness functions, e.g. the linear fitness functions, that are neither GA-deceptive nor GA-hard yet hard to optimization using genetic algorithms[13, 14]; for such smooth fitness functions, the fitness differences among individuals and schemata are so small that selection fails to choose the right ones with high probabilities, which degrades the GA's search.

Therefore, before discussing how to combine the superior building blocks to lead the search to the

desired regions, it is necessary first to clarify whether or not a representation can really present superior building blocks in the initial population.

In this paper, based on the previous work Ref. 15, the inherent periodicity in the search behavior of GA's is to be unveiled to show how to make use of this periodicity in the design of representation so as to get superior order-1 building blocks.

The rest of the paper is organized as follows. Section 2 gives some preliminaries, including the concept of locus factor as a measurement of fitness difference among order-1 schemata with the same fixed bit. The periodicity in the search behavior of genetic algorithms is introduced in Section 3. Section 4 shows how cardinality number m of the coding alphabet affects the supply of superior order-1 building blocks for single-period and multiple-period fitness functions with period m^{-i}. Simultation results for several typical cases are given in Section 5 to illustrate the effectiveness of the proposed method. Finally, Section 6 summarizes the paper.

2. Preliminaries (Fitness Difference and Locus Factor)

Let us consider a schema taken from the alphabet $\{*\} \cup A$, where * stands for "don't care" bit, and $A = \{0, 1, \cdots, m-1\}$, ($m \in Z^+$, $m > 1$), is the coding alphabet of the genetic algorithm, named as *m-allelic coding alphabet* for short. The number of the fixed bits of a schema is defined as the *order* of the schema. To simplify analysis, only schemata of order 1 are considered. A schema is called an order-1 schema at locus *i*, if its fixed position is at the i^{th} bit of the string counted from the rightmost. For example, 1***, **0* are order-1 schemata at the 4^{th} and 2^{nd} locus, respectively. Accordingly, a highly fit order-1 schema at the locus is named as a superior order-1 building block at locus *i*. A string is said to be a sample of a schema, or the latter is said to be matching the former, if the string is identical to the schema at the schema's fixed positions. Here and throughout, the average fitness of all the examples of an order-1 schema in the whole search space is defined as the fitness of this schema.

Without loss of generality, consider the fitness function $y = f(x)$, where $x \in [0, 1)$, $f(x) \geq 0$, and with at least one point whose fitness value is larger than 0. Assume that the coding is linear weighted with m-allelic coding alphabet, i.e., each solution of the search space, x, is encoded as $x = \sum_{i=1}^{l} x_i m^{i-l-1}$, where l is string length, and $x_i \in \{0, 1, \cdots, m-1\}$. For example, for the 3-allelic coding of string length 10, $x = \sum_{i=1}^{10} x_i 3^{i-11}$, and $x_i \in \{0, 1, 2\}$. Obviously, a superior order-1 building block at a certain locus indicates fitness variances among the order-1 schemata at this locus. In this paper, "locus factor" is introduced to describe the fitness variances.

Definition 1: The ratio of the maximal fitness to the average fitness of all the order-1 schemata at locus i is called the **locus factor** of this locus, denoted by θ_i, where $1 \leq i \leq l$.

The fitness of the schema *** x_i ***, denoted by $\bar{f}(x_i)$, is

$$\bar{f}(x_i) = \frac{\sum_{x_1=0}^{m-1} \cdots \sum_{x_{i-1}=0}^{m-1} \sum_{x_{i+1}=0}^{m-1} \cdots \sum_{x_l=0}^{m-1} f\left(x_i m^{i-l-1} + \sum_{j=1, j\neq i}^{l} x_j m^{j-l-1} \right)}{m^{l-1}}. \quad (1)$$

The average fitness of all the order-1 schemata at the i^{th} locus is

$$\bar{f}(L_i) = \frac{\sum_{x_i=0}^{m-1} \bar{f}(x_i)}{m}. \quad (2)$$

By Definition 1, the locus factor of the i^{th} locus is given by

$$\theta_i = \max_{x_i}[\bar{f}(x_i)/\bar{f}(L_i)] = \max_{x_i}[\bar{f}(x_i)]/\bar{f}(L_i). \quad (3)$$

Note that $\bar{f}(L_i) = \frac{1}{m}\sum_{x_i=0}^{m-1} \bar{f}(x_i) \leq \max_{x_i}[\bar{f}(x_i)] \leq \sum_{x_i=0}^{m-1} \bar{f}(x_i)$ $= m \cdot \bar{f}(L_i)$, therefore $1 \leq \theta_i \leq m$.

A simple example below illustrates the concept of locus factor. Suppose that the 3-allelic coding of string length 2 is used, and the fitness values of the strings are: $f(00) = 3$, $f(01) = 0$, $f(02) = 2$, $f(10) = 2$, $f(11) = 1$, $f(12) = 1$, $f(20) = 5$, $f(21) = 3$, $f(22) = 2$. From (3), one has $\bar{f}(0*) = (f(00) + f(01) + f(02))/3 = 1.67$, $\bar{f}(1*) = (f(10) + f(11) + f(12))/3 = 1.33$, and $\bar{f}(2*) = (f(20) + f(21) + f(22))/3 = 3.33$. The average fitness of all the order-1 schemata at locus 2 is $\bar{f}(L_2) = (\bar{f}(0*) + \bar{f}(1*) + \bar{f}(2*))/3 = 2.11$. Since $\bar{f}(2*)$ is the maximum among the three fitness values of the order-1 schemata at locus 2, according to (3), the locus factor at locus 2 is $\bar{f}(2*)/\bar{f}(L_2) = 1.58$. By Definition 1, if the locus factor of locus i is significantly larger than 1, there will

be a superior order-1 building block with fitness much higher than the other order-1 schemata at this locus. Here in this example, the superior building block is 2*.

3. Periodicity in GA's Search Behavior

In a GA's population, each individual can be viewed as a sampling point of the search space. The iterative mutation of the individual generates a serial of sampling points in the space. If a string bit-wisely mutates in such a manner that the allele value at a particular locus varies, while the allele values at the other loci remain unchanged, there will be a period among the serial of sampling points. For example, assume that a genetic algorithm is coded with the 3-allelic coding, i.e., $x = \sum_{i=1}^{4} 3^{i-5} x_i$. If the allele values at the 1st, 2nd, and 4th locus are all zeros, then the variation of the allele value at the 3rd locus results in strings 0000, 0100, and 0200, corresponding to decoded values, 0.000, 0.111, and 0.222, respectively. The period among these three points is 0.111. Obviously, the period depends on the position of the locus on the string. Another serial of sampling at the same locus may yield 1001, 1101, and 1201, and the period remains unchanged. However, the variation of the allele value at the 4th locus will result in a different period of 0.333. Moreover, the cardinality of the coding alphabet also affects the period. With the binary coding $x = \sum_{i=1}^{4} 2^{i-5} x_i$, mutation at the 4th locus results in 0000 and 1000, corresponding to 0.000 and 0.500, respectively, which indicates a period of 0.50. During the evolution, mutation can occur at different positions of the string, and therefore, a genetic algorithm samples the search space simultaneously with multiple sampling periods. For example, with the 3-allelic and binary coding given above, mutation at the 1st, 2nd, 3rd, and 4th loci corresponds to sampling periods of 3^{-4} , 3^{-3} , 3^{-2} , 3^{-1}, and 2^{-4} , 2^{-3} , 2^{-2} , 2^{-1}, respectively.

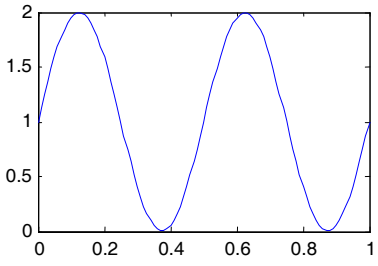

Fig.1.a. A sine function of period ½

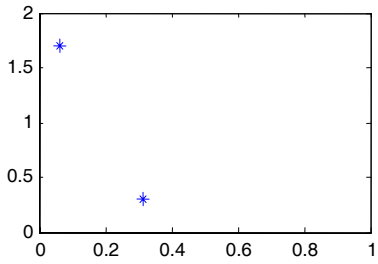

Fig.1.b. Sampling the sine function with period ¼ (the first two points)

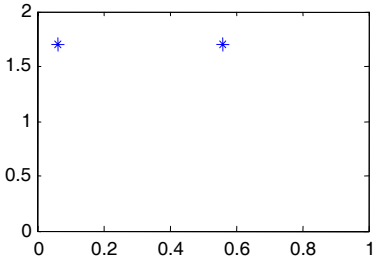

Fig.1.c. Sampling the sine function with period ½ (the first two points)

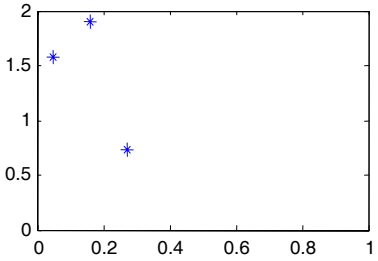

Fig.1.d. Sampling the sine function with period 1/9 (the first three points)

Intuitively, sampling with different periods extracts different features of a periodical function. As an example, Fig. 1. illustrates the differences of sampling results for $y = \sin 4\pi x + 1$ among different sampling periods. Fig.1.*b*., *c*., and *d*. give the first two/three points of the sampling with different periods. As is shown in Fig.1 *b*., one of the sampled points is significantly fitter than the other, while in Fig.1 *c*. and *d*., the differences of the fitness values between/among the sampled points are much smaller. Actually, if the binary coding is used, all the sampled points with higher fitness values are matched by the schema *0**. Consequently, *0** is much fitter than *1**, which, by Definition 1, indicates that the locus factor of the 3rd locus is significantly larger than 1. As for periods 2^{-1} and 3^{-2} , the differences among fitness values are so

Published by Atlantis Press

small that the locus factor of locus 2 of the binary coding and that of locus 3 of the 3-allelic coding cannot be large.

4. Function Period and Coding-Alphabet Cardinality

A straightforward suggestion from the periodicity behavior of the GA's is that, for a function with period m^{-i}, the m-allelic coding may generate large locus factor at locus i.

Consider a periodical function $y = f(x) = f(x \pm kT)$, where k is a positive integer, and $T = m^{-h}$, in which $h \in \{0, \cdots, l-1\}$. In terms of (1), one has

$$\bar{f}(x_{l-h}) = \frac{\sum_{x_1=0}^{m-1} \cdots \sum_{x_{l-h-1}=0}^{m-1} \sum_{x_{l-h+1}=0}^{m-1} \cdots \sum_{x_l=0}^{m-1} f(x_{l-h} m^{l-h-l-1} + \sum_{j \neq l-h}^{l} x_j m^{j-l-1})}{m^{l-1}}$$

$$= \frac{m^h \cdot \sum_{x_1=0}^{m-1} \cdots \sum_{x_{l-h-1}=0}^{m-1} f(x_{l-h} m^{-h-1} + \sum_{j=1}^{l-h-1} x_j m^{j-l-1})}{m^{l-1}}. \quad (4)$$

Further from (2), we have

$$\bar{f}(L_{l-h}) = \frac{\sum_{x_{l-h}=0}^{m-1} \bar{f}(x_{l-h})}{m}$$

$$= m^{h-l} \cdot \sum_{x_{l-h}=0}^{m-1} \left(\sum_{x_1=0}^{m-1} \cdots \sum_{x_{l-h-1}=0}^{m-1} f\left(x_{l-h} m^{-h-1} + \sum_{j=1}^{l-h-1} x_j m^{j-l-1} \right) \right). \quad (5)$$

Thus, the locus factor of locus $l-h$ is

$$\theta_{l-h} = \max_{x_{l-h}} [\bar{f}(x_{l-h})] / \bar{f}(L_{l-h})$$

$$= \frac{m \cdot \max_{x_{l-h}} \left(\sum_{x_1=0}^{m-1} \cdots \sum_{x_{l-h-1}=0}^{m-1} f\left(x_{l-h} m^{-h-1} + \sum_{j=1}^{l-h-1} x_j m^{j-l-1} \right) \right)}{\sum_{x_{l-h}=0}^{m-1} \left(\sum_{x_1=0}^{m-1} \cdots \sum_{x_{l-h-1}=0}^{m-1} f\left(x_{l-h} m^{-h-1} + \sum_{j=1}^{l-h-1} x_j m^{j-l-1} \right) \right)}. \quad (6)$$

When x_{l-h} varies from 0 to m-1, $x_{l-h} m^{-h-1} + \sum_{j=1}^{l-h-1} x_j m^{j-l-1}$ can be viewed as a sampling within the range of $[0, \ m^{-h}]$, i.e., $[0, \ T]$, with sampling period T/m. If the value of the periodical function varies significantly within a period, there will be at least one point whose fitness value is much larger than the average of the fitness values of all the points within this period. In this case, $\max_{x_{l-h}} \left(\bar{f}(x_{l-h}) \right) >> \sum_{x_{l-h}=0}^{m-1} \bar{f}(x_{l-h}) / m$, i.e., $\theta_{l-h} >> 1$.

Therefore, it is concluded that if the period of a non-negative function defined on [0, 1) is m^{-h}, where m and h are positive integers larger than 1, and the value of the function varies significantly within a period, then a large locus factor will be obtained at locus $l-h$ of the m-allelic-coded strings of length $l > h$.

Some samples are given in Tables 1, 2 and 3, in which the binary coding, the 3-allelic coding, and the 5-allelic coding are used, respectively. As shown in Table 1, the diagonal locus factors are much larger than the others. For each of the functions of period 2^{-k} ($k < 10$), the binary coding does result in a large locus factor at locus $10-k$. Similar results are also shown in Table 2 and Table 3, where large locus factors are gained on the diagonals of the tables.

As counterexamples, the locus factors of binary coding for the sine functions with periods 3^{-k} and 5^{-k}, respectively, are given in Table 4, in which most of the locus factors larger than 1.00 are at the 10th loci, rather than on the diagonal as shown in Tables 1, 2, and 3. Moreover, as compared with the large locus factors in Tables 1, 2, and 3, the large locus factors in Table 4 are much smaller, with the largest among them being merely 1.21.

Table 1. Locus factors of the binary coding of string length 10 for sine functions with periods 2^{-k}

Function	Allele number	10th	9th	8th	7th	6th	5th	4th	3rd	2nd	1st
$y = \sin 2\pi x + 1$	Locus factor	**1.64**	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
$y = \sin 4\pi x + 1$	Locus factor	1.00	**1.64**	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
$y = \sin 8\pi x + 1$	Locus factor	1.00	1.00	**1.64**	1.00	1.00	1.00	1.00	1.00	1.00	1.00
$y = \sin 16\pi x + 1$	Locus factor	1.00	1.00	1.00	**1.64**	1.00	1.00	1.00	1.00	1.00	1.00
$y = \sin 32\pi x + 1$	Locus factor	1.00	1.00	1.00	1.00	**1.64**	1.00	1.00	1.00	1.00	1.00
$y = \sin 64\pi x + 1$	Locus factor	1.00	1.00	1.00	1.00	1.00	**1.64**	1.00	1.00	1.00	1.00
$y = \sin 128\pi x + 1$	Locus factor	1.00	1.00	1.00	1.00	1.00	1.00	**1.64**	1.00	1.00	1.00
$y = \sin 256\pi x + 1$	Locus factor	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.64**	1.00	1.00

Table 2. Locus factors of the 3-allelic coding of string length 6 for sine functions with periods 3^{-k}

Function	Allele number	6^{th}	5^{th}	4^{th}	3^{rd}	2^{nd}	1^{st}
$y=\sin 2\pi x+1$	Locus factor	**1.72**	1.00	1.00	1.00	1.00	1.00
$y=\sin 6\pi x+1$	Locus factor	1.00	**1.72**	1.00	1.00	1.00	1.00
$y=\sin 18\pi x+1$	Locus factor	1.00	1.00	**1.72**	1.00	1.00	1.00
$y=\sin 54\pi x+1$	Locus factor	1.00	1.00	1.00	**1.72**	1.00	1.00
$y=\sin 162\pi x+1$	Locus factor	1.00	1.00	1.00	1.00	**1.72**	1.00
$y=\sin 486\pi x+1$	Locus factor	1.00	1.00	1.00	1.00	1.00	**1.72**

Table 3. Locus factors of the 5-allelic coding of string length 4 for sine functions with periods 5^{-k}

Function	Allele number	4^{th}	3^{rd}	2^{nd}	1^{st}
$y=\sin 2\pi x+1$	Locus factor	**1.89**	1.00	1.00	1.00
$y=\sin 10\pi x+1$	Locus factor	1.00	**1.89**	1.00	1.00
$y=\sin 50\pi x+1$	Locus factor	1.00	1.00	**1.89**	1.00
$y=\sin 250\pi x+1$	Locus factor	1.00	1.00	1.00	**1.89**

Table 4. Locus factors of the binary coding of string length 10 for sine functions with periods 3^{-k} and 5^{-k}

Function	Allele number	10^{th}	9^{th}	8^{th}	7^{th}	6^{th}	5^{th}	4^{th}	3^{rd}	2^{nd}	1^{st}
$y=\sin 6\pi x+1$	Locus factor	1.21	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
$y=\sin 18\pi x+1$	Locus factor	1.07	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
$y=\sin 54\pi x+1$	Locus factor	1.02	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
$y=\sin 162\pi x+1$	Locus factor	1.01	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
$y=\sin 10\pi x+1$	Locus factor	1.13	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
$y=\sin 50\pi x+1$	Locus factor	1.03	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
$y=\sin 250\pi x+1$	Locus factor	1.01	1.00	1.00	1.00	1.00	1.00	1.00	1.03	1.00	1.00

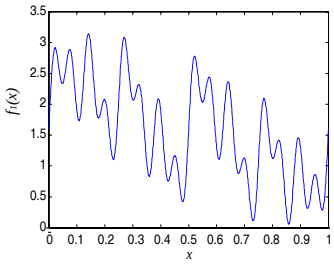

Fig. 2.a. Trajectory of $f_1(x)$

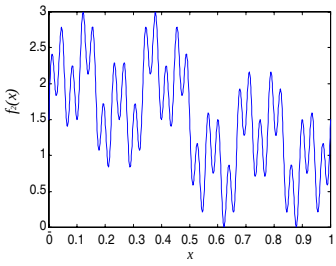

Fig. 2.b. Trajectory of $f_2(x)$

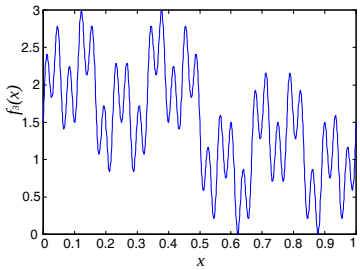

Fig. 2.c. Trajectory of $f_3(x)$

Table 5. Locus factors of the binary, 3-allelic, and 5-allelic coding employed to $f_1(x)$

Binary Coding	Allele number	10^{th}	9^{th}	8^{th}	7^{th}	6^{th}	5^{th}	4^{th}	3^{rd}	2^{nd}	1^{st}
	Locus factor	**1.199**	**1.199**	**1.199**	**1.199**	**1.199**	1.000	1.000	1.000	1.000	1.000
3-allelic Coding	Allele number	7^{th}		6^{th}	5^{th}	4^{th}		3^{rd}	2^{nd}		1^{st}
	Locus factor	**1.433**		1.000	1.001	1.000		1.000	1.000		1.000
5-allelic Coding	Allele number	5^{th}			4^{th}	3^{rd}			2^{nd}		1^{st}
	Locus factor	**1.506**			1.001	1.000			1.000		1.000

The above conclusion can be extended to a certain class of multiple-periodical functions. If the periods of a function are $T_1,\cdots,T_n$, where $T_i=m^{-h-i+1}$, $i=1,\cdots,n$, and $n\in Z^+$, and the value of the function varies greatly within each period, then the *m*-allelic coding is expected to generate large locus factors at loci, $l-h,\cdots,l-h-n+1$.

Consider functions $f_1(x)=0.5\sum_{k=0}^{4}\sin(2^k\cdot 2\pi x)+1.6$, $f_2(x)=0.5\sum_{k=0}^{3}\sin(3^k\cdot 2\pi x)+1.5$, and $f_3(x)=0.5\sum_{k=0}^{2}\sin(5^k\cdot 2\pi x)+1.5$, whose trajectories are shown

in Fig. 2. Since the value of a sine function varies significantly within a period, the coding schemes of proper alphabet cardinalities can produce large locus factors at particular loci. Here, the binary coding of string length 10, the 3-allelic coding of string length 7, and the 5-allelic coding of string length 5 are respectively used to represent the solutions of these functions. The corresponding locus factors for $f_1(x)$ are given in Table 5; and those for $f_2(x)$ and $f_3(x)$ are shown in Tables 6 and 7, respectively.

In Tables 5, 6, and 7, the locus factors are given in rows, and those larger than 1.1 are typed in boldface. As for $f_1(x)$ in Table 5, there are totally 5 large locus factors when the binary coding is used, while only one large locus factor when the other coding schemes are employed. Therefore, when applied to $f_1(x)$, the binary coding obtains more superior order-1 building blocks than the 3-allelic and 5-allelic coding. Similar results can be obtained for the other two functions, as shown in Tables 6 and 7, where the 3-allelic coding for $f_2(x)$ and the 5-allelic coding for $f_3(x)$ surpass the other coding schemes at achieving more large locus factors.

Table 6. Locus factors of the binary, 3-allelic, and 5-allelic coding employed to $f_2(x)$

Binary Coding	Allele number	10^{th}	9^{th}	8^{th}	7^{th}	6^{th}	5^{th}	4^{th}	3^{rd}	2^{nd}	1^{st}
	Locus factor	**1.314**	1.000	1.002	1.002	1.000	1.002	1.000	1.000	1.000	1.000
3-allelic Coding	Allele number	7^{th}		6^{th}	5^{th}		4^{th}	3^{rd}		2^{nd}	1^{st}
	Locus factor	**1.239**		**1.238**	**1.238**		**1.238**	1.000		1.000	1.000
5-allelic Coding	Allele number	5^{th}			4^{th}		3^{rd}		2^{nd}		1^{st}
	Locus factor	**1.381**			1.001		1.001		1.000		1.000

Table 7. Locus factors of the binary, 3-allelic, and 5-allelic coding employed to $f_3(x)$

Binary Coding	Allele number	10^{th}	9^{th}	8^{th}	7^{th}	6^{th}	5^{th}	4^{th}	3^{rd}	2^{nd}	1^{st}
	Locus factor	**1.263**	1.001	1.001	1.001	1.000	1.001	1.000	1.000	1.000	1.000
3-allelic Coding	Allele number	7^{th}		6^{th}	5^{th}		4^{th}	3^{rd}		2^{nd}	1^{st}
	Locus factor	**1.296**		1.000	1.000		1.001	1.000		1.000	1.000
5-allelic Coding	Allele number	5^{th}			4^{th}		3^{rd}		2^{nd}		1^{st}
	Locus factor	**1.297**			**1.297**		**1.297**		1.000		1.000

5. Simulations

In the simulations, genetic algorithms with the binary coding of string length 11, the 3-allelic coding of string length 7, and the 5-allelic coding of string length 5, respectively, are used to search for the maximal or sub-maximal values of the three functions given in Section 4. The algorithms employ roulette wheel and elitism selection, single-point crossover, and bitwise mutation. For the *m*-allelic coding, the mutation is to replace the original allele value with another integer randomly selected from $\{0, \cdots, m-1\}$. The population size, crossover probability, and mutation probability are 30, 0.8 and 0.005, respectively. For every function, 1000 calculations are performed using each of the three coding schemes in turn, with each calculation runs for 20 generations.

The Schema Theorem states that a superior building block receives an exponentially increasing trial in subsequent generations. As a result, the superior building blocks will soon dominate the population. Thus, we introduced "domination rate" as an indication of superior order-1 building blocks. An order-1 schema is said to dominating the final population if all the individuals in the final population are examples of the schema. In this paper, the domination rate of a particular order-1 schema stands for the rate of the times at which it dominates the final populations to the total number of the calculations. The results are given in Tables 8, 9, and 10, respectively. In the tables, the domination rates are listed for all the possible allele values at each locus, which stand for the domination rates of the corresponding order-1 schemata. High rates are highlighted in boldface.

For function $f_1(x)$ in Table 8, the allele value "0" at the 11^{th} locus, which stands for the order-1 schema 0**********, dominates the final populations for 790 times. The schemata, *0*********, **0********,

0****, and ****0******, also have very high domination rates. They are all superior building blocks, and each of them has much higher chance to survive than the other order-1 schema at the same fixed position. In contrast, only the schema 0********** dominates the final populations with high rates for $f_2(x)$ and $f_3(x)$. Similar results can be observed in Tables 9 and 10, in which the proper coding schemes generate more order-1 building blocks than the others.

We have also tried other positive integers as the coding alphabet cardinalities, e.g., 6-11, etc; the simulation results, which are omitted for sake of space limitation, were similar to those listed in Table 8-10, and supported the same conclusion, i.e., the *m*-allelic coding provides more superior order-1 building blocks for multiple-periodical fitness function with periods satisfying $T_i = m^{-h-i+1}$, $i = 1, \cdots, n$ than the coding with other alphabet cardinalities.

Table 8. Domination rate of the allele values at the loci of the binary strings

Function	Allele value	11th	10th	9th	8th	7th	6th	1st-5th
$f_1(x)$	0	**0.790**	**0.586**	**0.527**	**0.783**	**0.298**	0.417	0.30 < Each < 0.50
	1	0.039	0.131	0.276	0.080	0.004	0.470	0.30 < Each < 0.50
$f_2(x)$	0	**0.998**	0.300	0.289	0.387	0.408	0.421	0.30 < Each < 0.50
	1	0.001	0.287	0.356	0.423	0.381	0.440	0.30 < Each < 0.50
$f_3(x)$	0	**0.958**	0.332	0.368	0.412	0.409	0.417	0.30 < Each < 0.50
	1	0.003	0.390	0.445	0.360	0.364	0.398	0.30 < Each < 0.50

Table 9. Domination rate of the allele values at the loci of the 3-allelic strings

Function	Allele value	7th	6th	5th	4th	3rd	2nd	1st
$f_1(x)$	0	**0.820**	0.132	0.327	0.198	0.256	0.221	0.187
	1	0.033	0.325	0.329	0.198	0.263	0.189	0.203
	2	0.003	0.267	0.120	0.320	0.228	0.231	0.276
$f_2(x)$	0	**0.468**	**0.364**	**0.400**	**0.541**	0.223	0.237	0.267
	1	**0.253**	**0.387**	**0.356**	**0.169**	0.223	0.237	0.149
	2	0.001	0.010	0.005	0.003	0.249	0.210	0.200
$f_3(x)$	0	**0.723**	0.130	0.324	0.126	0.227	0.198	0.201
	1	0.065	0.150	0.147	0.198	0.210	0.253	0.241
	2	0.001	0.376	0.168	0.304	0.205	0.198	0.195

Table 10. Domination rate of the allele values at the loci of the 5-allelic strings

Functions	Allele value	5th	4th	3rd	2nd	1st
$f_1(x)$	0	**0.234**	0.007	0.012	0.033	0.038
	1	**0.162**	**0.169**	0.021	0.056	0.101
	2	0.001	0.001	0.067	0.048	0.033
	3	0.002	0.078	0.069	0.089	0.041
	4	0.000	0.002	0.077	0.081	0.065
$f_2(x)$	0	**0.186**	0.013	0.077	0.073	0.058
	1	**0.115**	0.044	0.041	0.053	0.049
	2	0.023	0.021	0.059	0.043	0.022
	3	0.007	0.120	0.035	0.044	0.098
	4	0.002	0.066	0.038	0.054	0.075
$f_3(x)$	0	0.066	0.081	0.132	0.047	0.042
	1	**0.231**	**0.238**	**0.295**	0.071	0.036
	2	0.077	0.043	0.021	0.045	0.067
	3	0.002	0.004	0.001	0.067	0.101
	4	0.004	0.002	0.003	0.045	0.059

6. Conclusions

A genetic algorithm samples the search space simultaneously with multiple sampling periods determined by the cardinality of the coding alphabet. Thus, a coding scheme with cardinality in accordance with the periods of the fitness function is expected to present more superior order-1 building blocks than those with improper cardinalities. It is shown that if the periods of the fitness function satisfy $T_i = m^{-h-i+1}$, $i = 1, \cdots, n$, and the value of the function varies greatly within each period, then the *m*-allelic coding can generate large locus factors at loci, $l-h, \cdots, l-h-n+1$, and may surpass the coding with other alphabet cardinalities in the supply of superior order-1 building blocks.

Given an arbitrary function, its main frequency components can hardly be expressed as m^{h+i-1}. Therefore, it should be interesting to find out whether or not rare superior order-1 building blocks can be generated for such functions, as long as a linear-weighted coding $x = \sum_{i=1}^{l} x_i m^{i-l-1}$ is employed.

Acknowledgements

The research was supported by SRF for ROCS, SEM, China, and BK21, Korea.

The authors would also like to thank Prof. Yunong Zhang, Dr. Huang Zhong, Mrs. Ming Xiao, and Dr. Xuelin Wu for the fruitful discussions.

References

1. J.H. Holland, *Adaptation in Natural and Artificial systems* (Ann Arbor, MI: University of Michigan Press, 1975).
2. D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning* (Reading, MA: Addison Wesley, 1989).
3. D.E. Goldberg, K. Sastry, and T. Latoza, On the supply of building blocks, in *Proc. of the 2001 Genetic and Evolutionary Computation Conference*, eds. L. Spector. (San Francisco, Calif., Kaufmann, 2001), pp. 336-342.
4. W.A. Chang and S.R. Rudrapatna, Building-block supply in real-coded genetic algorithms: a first step on the population-sizing model, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* E89-A(7) (2006) 2072-2078.
5. C. Kahraman et al., An application of effective genetic algorithms for solving hybrid flow shop scheduling problems, *International Journal of Computational Intelligence Systems* 1(2) (2008) 134-147.
6. S. Forrest, and M. Mitchell, Relative building-block fitness and the building-block hypothesis, in *Foundations of Genetic Algorithms 2*, eds. D. Whitley (1993), pp. 109-126.
7. A.S. Wu, and K.S. DeJong, An examination of building block dynamics in different representations, in *Proc. of the 1999 Congress on Evolutionary Computation* (1999), pp. 6-9.
8. D.E. Goldberg, Construction of high-order deceptive functions using low-order Walsh coefficients, *Annals of Mathematics and Artificial Intelligence* 5(8) (1992) 35-48.
9. K. Deb, J. Horn, and D.E. Goldberg, Multimodal deceptive functions, *Complex Systems* 7(2) (1993) 131-153.
10. K. Deb and D.E. Goldberg, Sufficient conditions for deceptive and easy binary functions, *Annals of Mathematics and Artificial Intelligence* 7(10) (1994) 385-408.
11. T. Kuo and S.Y. Hwang, Genetic algorithm with disruptive selection, *IEEE Trans. Systems, Man, and Cybernetics, Part B* 26(2) (1996) 299-307.
12. C. van Hoyweghen, D.E. Goldberg, and B. Naudts, Building block superiority, multimodality and synchronization problems, in *Proc. of the 2001 Genetic and Evolutionary Computation Conference*, eds. L. Spector, (San Francisco, Calif., Kaufmann, 2001), pp. 694-701.
13. H.Q. Mo, F. Luo, X.M. Hou, and Z.Y. Mao, GA-easy does not imply GA fast optimization, *Computer Engineering and Applications* 38(3) (2002) 20-22 (in Chinese).
14. H.Q. Mo, X.Y. Li, G.C. Wan, and X. Tian, On the limitation of linear-weighted-coded genetic algorithms in the optimization of linear functions, *Computer Engineering and Applications* 43(8) (2007) 85-87 (in Chinese).
15. H.Q. Mo, J.B. Park, and Y.H. Joo, Locus factor and function period, in *Proceedings of the First International Conference on Machine Learning and Cybernetics*, (Beijing, 2002), pp. 1951-1955.