Adaptive Group Mutation for Tackling Deception in Genetic Search

SHENGXIANG YANG
Department of Computer Science
University of Leicester
University Road, Leicester LE1 7RH
UNITED KINGDOM
s.yang@mcs.le.ac.uk http://www.cs.le.ac.uk/~syang

Abstract: In order to study the efficacy of genetic algorithms (GAs), a number of fitness landscapes have been designed and used as test functions. Among these functions a family of deceptive functions have been developed as difficult test functions for comparing different implementations of GAs. In this paper an adaptive group mutation (AGM), which can be combined with traditional bit mutation in GAs, is proposed to tackle the deception problem in genetic searching. Within the AGM, those genes that have converged to certain threshold degree are adaptively grouped together and subject to mutation together with a given probability. To test the performance of the AGM, experiments were carried out to compare GAs that combine the AGM and GAs that use only traditional bit mutation with a number of suggested "standard" fixed mutation rates over a set of deceptive functions as well as non-deceptive functions. The results demonstrate that GAs with the AGM perform better than GAs with only traditional bit mutation over deceptive functions and as well as GAs with only traditional bit mutation over non-deceptive functions. The results show that the AGM is a good choice for GAs since most problems may involve some degree of deception and deceptive functions are difficult for GAs.

Key-Words: Genetic algorithm, adaptive group mutation, bit mutation, deceptive functions, building blocks.

1 Introduction

As a class of optimum-seeking algorithms based on principles of natural evolution, genetic algorithms (GAs) maintain a population of individuals. Each individual is associated a fitness value according to a given problem objective function. GAs iteratively generate new population by selecting relatively fit individuals of the present population and performing recombination and mutation operations on these individuals [8]. Due to the properties of robustness and adaptability, GAs have been successfully applied to solve many hard optimization problems in both artificial and realistic. Yet GAs do fail.

In order to study the efficacy of GAs, a number of fitness landscapes have been designed and used as test functions. Among these functions a family of fitness landscapes are called *deceptive functions*, which are developed to challenge the building block hypothesis. In order to analyze how GAs work, Holland [11] introduced the concept of *schema* to describe a subset of binary strings that have similarities at certain positions and worked out the *schema theorem*. The schema theorem states that short, low-order, better than average schemas (or building blocks) receive an exponentially increasing number of trials in subsequent generations of a GA. A direct result of the schema theorem is the *building block hypothesis* that suggests that GAs work by combining low-order building blocks to form higher-order building blocks. The building block hypothesis is the fundamental working mechanism of GAs. Therefore, if in a function the low-order building blocks do not combine to form higher-order building blocks, GAs may have difficulty in solving this function. Deceptive functions are such family of functions where there exist low-order building blocks that do not combine to form higher-order building blocks: instead they form building blocks resulting in a deceptive solution that is sub-optimal itself or near a sub-optimal solution [15]. Deceptive functions are quite general. In fact, most problems may involve some degree of deception because one would not expect that all low-order building blocks will be consistent with relevant higher-order building blocks.

Since deceptive functions are supposedly difficult for GAs, considerable effort has been made to understand how to modify simple GAs to solve such difficult fitness landscapes [4, 9]. In this paper, an adaptive group mutation (*AGM*), which can be combined with traditional bit mutation in GAs, is proposed to tackle the deception problem in genetic searching. Within the AGM, those genes that have converged to certain threshold degree are adaptively grouped together and subject to mutation together with a given group mutation probability. The idea behind the AGM is to set up a channel in the search space for the GA to escape from the deceptive solution to the global optimal solution.

2 Deception in Genetic Search

In order to better understand the situations that are likely to create difficulty for GAs Goldberg [7] first introduced the concept of *deception*. Thereafter, many researchers have developed partially and fully deceptive functions [5, 9, 12]. In fully deceptive functions all low-order building blocks are deceptive with respect to the global optimal solution. The motivation of developing such functions is to create difficult test functions for comparing different GAs. The existence of deceptive building blocks in deceptive functions makes it difficult for GAs to search the global optimal solution. It is even claimed that the only challenging problems for GAs are problems that involve some degree of deception [15].

Ackley [1] first introduced trap functions that are defined in terms of *unitation* (the number of 1's in the string). A function of unitation has the same function value for all strings of identical unitation. In an L-bit unitation function, there are $L+1$ different function values. An L-bit trap function $f(u)$ is defined as a function of unitation *u* as follows:

$$f(u) = \begin{cases} \frac{a}{z} * (z-u), & if\ u \le z \\ \frac{b}{L-z} * (u-z), & otherwise \end{cases} \quad (1)$$

where a and b are the function values of the deceptive and global optimal solutions respectively, the parameter z is the location where the unitation search space of the function is divided into two basins: one leading to the global optimal solution and other leading to the deceptive solution. Fig. 1 shows a 4-bit trap function with $a = 0.6$, $b = 1.0$, and $z = 3$.

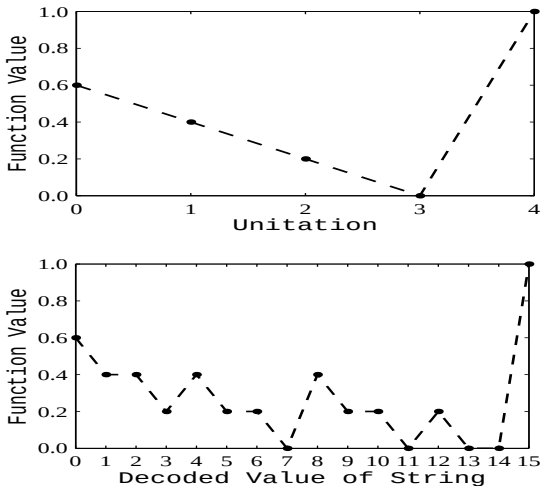

Fig.1 A 4-bit trap function with $a = 0.6$, $b = 1.0$, and $z = 3$ as a function of unitation (*Top*) and as a function of decoded value of binary strings (*Bottom*).

By explicitly calculating and comparing all schema fitness values, Goldberg [10] devised a 3-bit minimum fully deceptive problem as follows:

f(000) = 28, f(001) = 26, f(010) = 22, f(011) = 0
f(100) = 14, f(101) = 0, f(110) = 0, f(111) = 30

where all the order-1 and order-2 building blocks (e.g., "0**" and "*00") in the search space are deceptive and will lead the genetic search away from the global optimum "111" and instead toward the local optimum "000". Using an algorithm of constructing fully deceptive function, Whitley [15] developed a 4-bit fully deceptive problem as follows:

f(0000)= 28, f(0001)= 26, f(0010)= 24, f(0011)=18
f(0100)= 22, f(0101)= 16, f(0110)= 14, f(0111)= 0
f(1000)= 20, f(1001)= 12, f(1010)= 10, f(1011)=2
f(1100)= 8, f(1101)= 4, f(1110)= 6, f(1111)=30

As can be seen from above examples, a deceptive function usually has at least two optimal solutions: one global optimal solution and one local sub-optimal solution. The local solution is the best solution in its neighbourhood but inferior to the global solution and is usually known as the deceptive solution or *deceptive attractor* (by Whitley [15]) of the deceptive function. The deceptive attractor consists of deceptive building blocks and is usually a stable point in the search space of a deceptive function. The deceptive attractor need not be a local optimum in Hamming space. However, it has been shown by Whitley [15] that in a fully deceptive function the deceptive solution can be at most one-bit dissimilar to a local optimal solution in Hamming distance. Whitley has also proved the *deceptive attractor theorem* that states that in a fully deceptive function the deceptive attractor must have a complementary bit pattern to that found in the binary representation of the global optimum solution. The deceptive attractor theorem gives us a hint that the deception problem for GAs may be tackled by building up a bridge from the deceptive attractor to the global solution in the search space. This is where the adaptive group mutation comes in this paper.

3 Tackling Deception with Adaptive Group Mutation

According to Whitley's deceptive attractor theorem, the deceptive solution must be the complement of the global optimum in Hamming space. This information can be of practical significance. As discussed in [15], if deception occurs in a known or predictable location remapping strategies that move the global solution closer in Hamming space to the deceptive attractor

can be used to reduce the level of deception. However, the problem with remapping strategies is that in general it is not practical to know the exact location of deception in deceptive functions. Finding where deception locates is as difficult as optimizing the deception function. This also precludes such simple fixes for deception as inverting the final solution or inverting strings during the search and evaluating their complements.

But when the deception problem is considered more deeply we may find a way around the need of explicitly locating where deceptive building blocks reside. Let's look at the dynamic behaviour of deception in genetic search. Because deceptive schemas that represent the deceptive solution have better fitness than any other competitor schemas including that representing the global solution, GAs process them favourably in early generations. Solutions with these schemas dominate the population and GAs will finally be misled to the deceptive solution instead of the global optimal solution. The dynamic appearance in genotype will be that during early generations of search the alleles of those bits where deception occurs will converge to values that are consistent with the bit pattern of deceptive building blocks. Hence, we can use simple statistics of allele distribution in genotype to locate deceptive building blocks and group those bits converged to certain level together to be mutated in a whole with certain probability.

In this paper we assume that binary encoding is used for GAs. Due to the property of symmetry we can use the frequency of 1's in the alleles on a locus over the population to calculate the degree of convergence of that locus. Let $f_1(i,t)$ ($i = 1 \ldots L$ where *L* denotes the string length through this paper) denote the frequency of 1's in the alleles on locus *i* over the population at generation *t* and $d_c(i,t)$ denote the degree of convergence on locus *i* at generation *t*. Then $d_c(i,t)$ can be calculated from $f_1(i,t)$ as follows:

$$d_c(i,t) = \max\{f_1(i,t),\ 1 - f_1(i,t)\} \qquad (2)$$

Now during the progress of the GA, after a new population *t* has been generated we first calculate the frequency of 1's $f_1(i,t)$ on each locus *i* and from this the degree of convergence $d_c(i,t)$ is calculated for each locus *i*. Then, those loci that have the degree of convergence $d_c(i,t)$ going up to a given threshold level d_T are grouped together. And then, after the population *t* has undergone the crossover operation, it will undergo mutation operations as follows. Each individual *x* is first subjected to the group mutation with a given group mutation probability p_g. If a randomly created number $r = rand(0,1)$ is less than p_g, *x* will undergo the group mutation and those alleles that correspond to grouped loci will be inverted to their complement values while those not grouped loci will keep their alleles intact. However, if $r \geq p_g$, *x* will not undergo the group mutation: instead it will undergo the traditional bit mutation with a probability p_m. The pseudo-code for AGM is shown below, where $P'(t)$ is the intermediate population after crossover at generation *t*.

Procedure *AGM*:
begin
 Calculate $f_1(i,t)$ and $d_c(i,t)$ for locus *i* at gen. *t*;
 for $i := 1$ **to** *L* **do** {group loci by a mask vector G}
  if $d_c(i,t) \geq d_T$ **then** $G(i,t) := 1$;
  else $G(i,t) := 0$;
 endfor;
 for each member $x = (x_1 x_2 \ldots x_L) \in P'(t)$ **do**
  if $r = rand(0,1) < p_g$ **then** {group mutation on *x*}
   for $i := 1$ **to** *L* **do**
    if $G(i,t) = 1$ **then** $x_i := 1 - x_i$;
   endfor;
  else
   Perform traditional bit mutation on *x* with p_m;
 endfor;
end;

4 Computer Experiment Study

4.1 Design of Experiments

To test the performance of the AGM, in this study we constructed two deceptive functions: DF_1 that contains 10 copies of Goldberg's 3-bit fully deceptive subfunction (see Section 2) and DF_2 that contains 10 copies of Whitley's 4-bit fully deceptive subfunction (see Section 2). The optimal solutions for both DF_1 and DF_2 have a fitness of 300. We also constructed a *Trap Function* that contains 10 copies of Ackley's 4-bit trap function (see Fig. 1) and has an optimal fitness of 10.

Experiments were carried out to compare GAs that combine the AGM with traditional bit mutation and GAs that use only traditional bit mutation. To better understand the effect of the AGM, the mutation probability p_m for the bit mutation was varied from a series of recommended "standard" values: *1/L* by

Mühlenbein [14], $1.75/(N*\sqrt{L})$ by Bäck [3] where N is the population size, and 0.001 by De Jong [6]. And the 2-point crossover with a fixed crossover probability $p_c = 0.6$ and the 0.5 uniform crossover were chosen as crossover operators for GAs. Within the AGM, the group mutation probability p_g and the threshold convergence degree d_T were set to 0.01 and 0.85 respectively. For all the GAs, the fitness proportionate selection with the Stochastic Universal Sampling [2] and elitist model [6] were used and the population size N was set to 100.

For each experiment of combining the test function and the GA with different crossover and mutation, 100 independent runs were executed. In order to have a strict comparison the same 100 different random seeds were used to generate initial populations for the 100 runs of each experiment. For each run, the initial population was randomly created using a technique that generates exactly equal number of 0s and 1s for each locus, that is, $f_1(i,t) = 0.5$ for each locus i ($i = 1 \ldots L$) in the initial population. In this way the random sampling bias in the initial population (e.g., for some locus j, $f_1(j,0) = 0.9$ or 0.1 and hence $d_c(j,0) = 0.9$) that may misleads the AGM is cancelled. For each run, the best-so-far fitness was recorded every 100 evaluation[1] and the maximum allowable number of evaluations was set to 20000. Each experiment result was averaged over 100 independent runs.

4.2 Experiment Results and Discussions

The experiment results on different deceptive functions are shown in Fig. 2 through Fig. 4 (where BM means traditional bit mutation) respectively. From these figures, it can be seen that in general GAs with the AGM perform better than GAs without the AGM on the three deceptive functions. GAs with the AGM perform greatly better than GAs without the AGM on the deceptive functions when the mutation probability p_m for traditional bit mutation is set to $1.75/(N*\sqrt{L})$ or 0.001. For example, on DF_1 and with 2-point crossover, when $p_m = 1.75/(N*\sqrt{L})$ the GA with the AGM found the global optimal solution 86 times out of 100 runs while the GA without the AGM only succeeded 31 times and when $p_m = 0.001$ the GA with the AGM found the global optimal solution 81 times out of 100 runs while the GA without the AGM only succeeded 5 times. When

[1] Here, only those chromosomes changed by crossover and mutation operations were evaluated and counted into the number of evaluations.

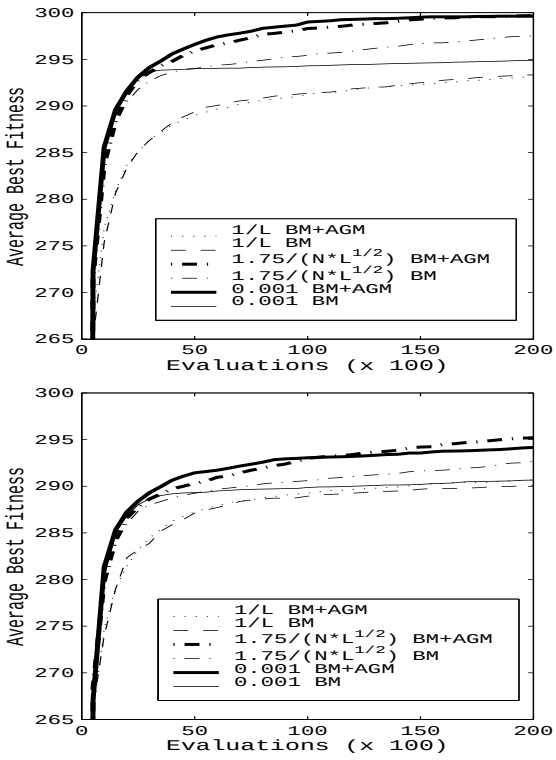

Fig.2 Average best-so-far fitness against evaluations of GAs with 2-Point Crossover (*Top*) and Uniform Crossover (*Bottom*) on Deceptive Function DF_1.

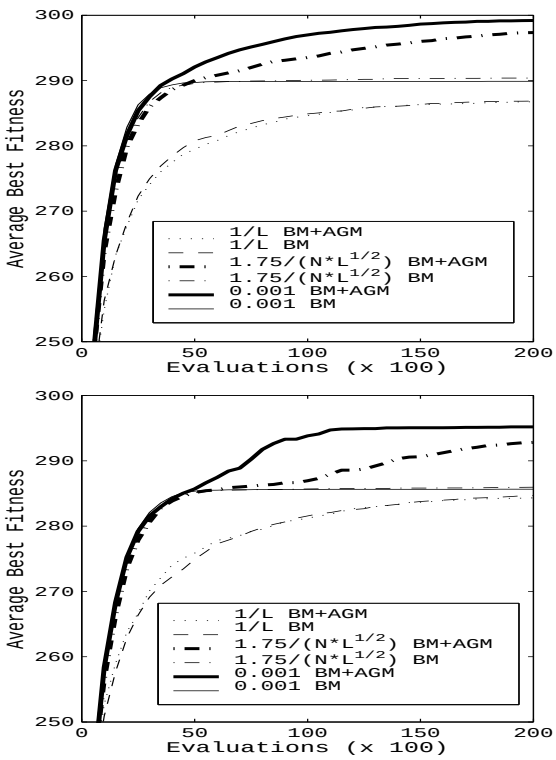

Fig.3 Average best-so-far fitness against evaluations of GAs with 2-Point Crossover (*Top*) and Uniform Crossover (*Bottom*) on Deceptive Function DF_2.

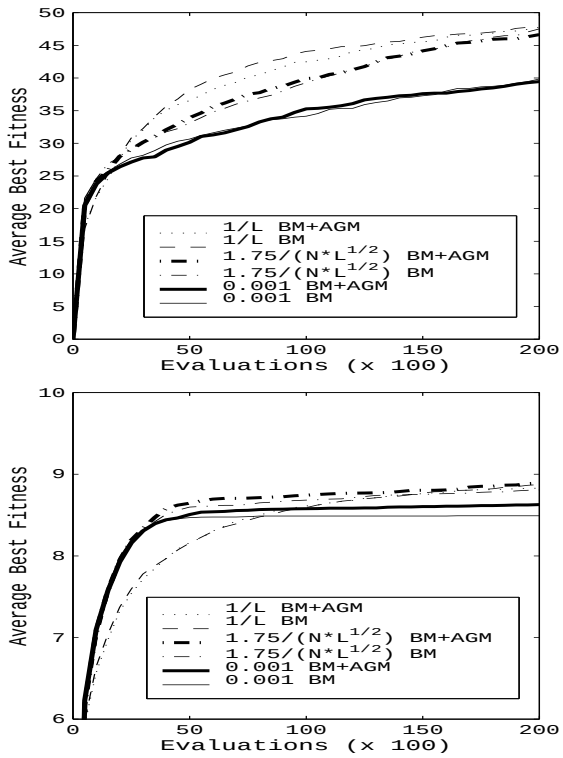

Fig.4 Average best-so-far fitness against evaluations of GAs with 2-Point Crossover (*Top*) and Uniform Crossover (*Bottom*) on Trap Function.

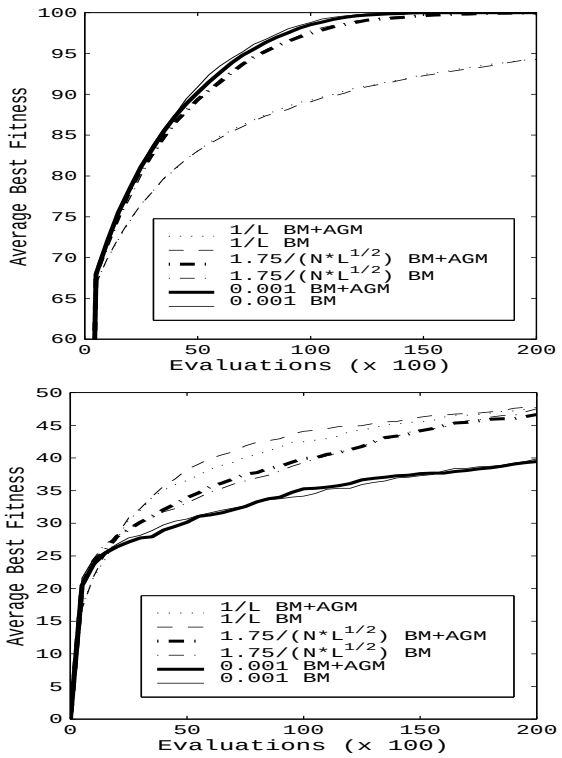

Fig.5 Average best-so-far fitness against evaluations of GAs with 2-Point Crossover on One-Max (*Top*) and Royal Road Function R_1 (*Bottom*).

$p_m = 1/L$ it seems that the AGM has little effect on the performance of GAs. The reason lies in that $p_m = 1/L$ is quite large for the domain of the three deceptive functions: $1/L = 0.033$ for DF_1 and the Trap Function and $1/L = 0.025$ for DF_2. This large setting of p_m heavily slows down the building up of deceptive building blocks as well as non-deceptive building blocks.

4.3 Experiments on Non-Deceptive Problems

Above experiment results showed the benefit of combining the AGM within GAs on deceptive functions. How about the effect of combining the AGM within the GA on non-deceptive functions? According to the "No Free Lunch" theorem [16], one might expect that the introduction of the AGM may degrade the performance of GAs on non-deceptive functions since the AGM may invert and hence destruct non-deceptive building blocks found so far as well. To test this further experiments were carried out on two typical non-deceptive functions: the One-Max problem introduced by Ackley [1] and the Royal Road function R_1 devised by Mitchell, Forrest and Holland [13]. The One-Max problem simply counts the 1's in a binary string as its fitness. The aim is to maximize ones in a string. A string length of 100 bits was used for this study. The Royal Road Function R_1, based on 64-bit binary strings, contains 8 disjunctive order-8 tailor-made building blocks (schemas). Each schema has 8 adjacent ones and contributes 8 to the fitness of a string when it exists in the string. The fitness of a string *x* is the summation of contributions from each schema, of which *x* is an instance. The optimal solution $x^* = 11\ldots1$ for R_1 has a fitness of 64.

The experiment results of GAs with 2-point crossover on the two non-deceptive problems are shown in Fig. 5. The experiment results of GAs with uniform crossover are similar and not shown here. From Fig. 5 it can be seen that GAs with the AGM perform as well as GAs without the AGM on these two problems. It seems that the introduction of the AGM doesn't affect the performance of the GA greatly if there is any effect. This is because the group mutation probability within the AGM can be set to a small value, e.g., $p_g = 0.01$ in this study. This means that a small portion of individuals in the population (statistically 1 out of 100) will actually undergo the group mutation. On non-deceptive functions this small portion of individuals that undergo the group mutation operations will usually be discarded during the next selection since their non-deceptive building

blocks are destructed and hence their fitnesses are decreased. However, on deceptive functions this small portion of group mutations can be of significant effect because they may invert deceptive building blocks to correct building blocks or invert the deceptive solution directly to the global optimal solution.

5 Conclusions

In this paper, an adaptive group mutation (AGM) is proposed with the aim to tackle the deception problem in genetic search. The AGM is an adaptive mutation operator that uses the statistics information from the search process as feedback information to adaptively group those loci that have currently converged to certain degree and then give these grouped loci chance to mutate together with a given group mutation probability.

To test the performance of the AGM, experiments were carried out to compare GAs that combine the AGM and GAs that use only traditional bit mutation with a number of suggested "standard" fixed mutation rates on a set of deceptive functions as well as non-deceptive functions. The experiment results demonstrate that GAs with the AGM perform better than GAs without the AGM on deceptive functions while perform as well as GAs without the AGM on non-deceptive functions though the AGM is aimed at tackling the deception problem in genetic search. The experiment results indicate that the AGM is a good adaptive mutation operator for GAs since deceptive functions are quite general and difficult for GAs.

Within the AGM, besides the mutation probability p_m for traditional bit mutation, there are other two key parameters: the threshold degree of convergence d_T and the group mutation probability p_g. The value of d_T controls when gene loci should be grouped, i.e., when the AGM should take action, while the value of p_g is used to control how deeply the AGM should take action. In this study, both d_T and p_g are set to fixed values. However, adapting their settings with the progress of searching may further improve GA's performance, which is one future work about the AGM. Combining the AGM with other adaptation techniques for GAs is another future work about the AGM.

References:

[1] D. H. Ackley, *A Connectionist Machine for Genetic Hillclimbing*, Boston, MA: Kluwer Academic Publishers, 1987.

[2] J. E. Baker, Reducing Bias and Inefficiency in the Selection Algorithms, *Proc.of the 2nd Int. Conf. on Genetic Algorithms*, J. J. Grefenstelle Ed., 1987, pp. 14-21, Lawrence Erlbaum Associates.

[3] T. Bäck, Self-Adaptation in Genetic Algorithms. *Proc. of the 1st European Conf. on Artificial Life*, F. J. Varela and P. Bourgine Eds., 1992, pp. 263-271, MIT Press.

[4] K. Deb, *Binary and Floating-point Function Optimization Using Messy Genetic Algorithms*, PhD Thesis, University of Alabama, USA, 1991.

[5] K. Deb and D. E. Goldberg, Sufficient Conditions for Arbitrary Binary Functions. *Ann. Math. Artif. Intellegence*, Vol. 10, 1994, pp. 385-408.

[6] K. A. De Jong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, PhD Thesis, University of Michigan, Ann Abor, 1975.

[7] D. E. Goldberg, Simple Genetic Algorithms and the Minimal, Deceptive Problem, *Genetic Algorithms and Simulated Annealing*, L. Davis Ed., 1987, pp. 74-88, Morgan Kaufmann.

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

[9] D. E. Goldberg, B. Korb and K. Deb, Messy Genetic Algorithms: Motivation, Analysis, and First Results. *Complex Systems*, Vol. 4, 1989, pp. 415-444.

[10] D. E. Goldberg, Genetic Algorithms and Walsh Functions: Part I, a Gentle Introduction. *Complex Systems*, Vol. 3, 1989, pp. 129-152.

[11] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

[12] G. E. Liepins and M. D. Vose, Representation Issues in Genetic Optimization. *J. Exp. Theor. Artificial Intelligence*, Vol. 2, 1990, pp. 4-30.

[13] M. Mitchell, S. Forrest and J. H. Holland, The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance. *Proc. of the 1st European Conference on Artificial Life*, F. J. Varela and P. Bourgine Eds., 1992, pp. 245-254.

[14] H. Mühlenbein, How Genetic Algorithms Really Work: I. Mutation and Hillclimbing. *Proc. of the 2nd Conf. on Parallel Problem Solving from Nature*, R. Männer and B. Manderick Eds., 1992, pp. 15-29.

[15] L. D. Whitley, Fundamental Principles of Deception in Genetic Search. *Foundations of Genetic Algorithms 1*, G. J. E. Rawlins Ed., 1991, pp. 221-241. Morgan Kaufmann Publishers.

[16] D. Wolpert and W. Macready, No Free Lunch Theorems for Optimization, *IEEE Trans. on Evolutionary Computation*, Vol. 1, No. 1, 1997, pp. 67-82.