The Role of Neutral and Adaptive Mutation in an Evolutionary Search on the OneMax Problem

Tina Yu

ChevronTexaco Information Technology Company
San Ramon, CA 94583, U.S.A.
tiyu@chevrontexaco.com

http://www.improvise.ws

Julian F. Miller

University of Birmingham
Birmingham, B152TT, U.K.
j.miller@cs.bham.ac.uk

http://www.cs.bham.ac.uk/~jfm

Abstract

We investigate neutrality in the simple Genetic Algorithms (SGA) and in our neutrality-enabled evolutionary system using the OneMax problem. The results show that with the support of limited neutrality, SGA is less effective than our system where a larger amount of neutrality is supported. In order to understand the role of neutrality in evolutionary search of this unimodal landscape, we have created a theoretical framework that gives the number of gene changes under different levels of neutrality. The interim results of this theoretical work are also presented.

1 INTRODUCTION

OneMax was devised by Ackley as one of the benchmark problems to test the generality of his stochastic iterated genetic hillclimbing search algorithm (Ackley, 1987). This problem has a simple formula: for a binary string x of length l, the problem is to maximize:

$$\sum x_i \quad , \quad x_i \in \{0, 1\}^l.$$

There is only one optimum solution in this problem: the binary string with all bits values equal to 1. This unimodal search space (see Section 5) is most suitable for hill climbing search algorithms. As expected, he reported that hill climbing methods outperform genetic and simulated annealing search methods on this problem.

Although not an ideal problem to demonstrate the strength of Genetic Algorithms (GAs) (Holland, 1975), OneMax has been used by many researchers to study different aspects of GAs because of its simplicity. For example, (Bäck, 1992) studied optimal mutation rates and their interaction with selection and self-adaptation on this problem, while (Giguere and Goldberg, 1998) used this problem to investigate the sampling size and selection scheme in GAs. The purpose of these studies was to gain understanding of how GAs perform search, not to assess the search ability of GAs against other methods.

In this work, we investigate solving the OneMax problem using an evolutionary system that supports neutrality. The goal of this study is twofold:

- To assess the generality of the neutrality-enabled evolutionary system.
- To understand the search behavior of the neutrality-enabled evolutionary system on this problem.

Previously, we have devised an evolutionary system whose genotype representation contains extra inactive genes (see Section 3). Mutations acting on these inactive genes have a neutral effect on the genotype's fitness. However, neutral mutations can change the dynamics of the evolutionary search process. When applied to a Boolean function and four needle-in-haystack problems, the results show that higher search success rates were obtained when a higher amount of neutral mutations were permitted during search (Yu and Miller, 2001; Yu and Miller, 2002). These results encourage us to test the system on different kinds of problems to assess its generality.

The search behavior of the neutrality-enabled evolutionary system has been analyzed quantitatively using the ratio between active and inactive gene changes (see Section 2). Previously, we carried out empirical studies of these ratios. In this work, we attempt a theoretical approach by formulating these ratios mathematically. Although this is a challenging task, our interim results indicate that it is achievable for the simple OneMax landscape (see Section 8).

The paper is organized as follows. Section 2 provides the background of this work. It first explains neutral theory in evolutionary biology and then gives a summary of our previous work on using neutrality for evolutionary search. Section 3 describes the neutrality-enabled evolutionary system. In Section 4 related work is reviewed. Section 5 analyzes the OneMax search space. In Section 6, the experimental setup is given. The results are presented in Section 7. Section 8 provides a theoretical study of the active and inactive gene changes during search. Finally, Section 9 gives our conclusions.

2 BACKGROUND

The theory of natural evolution established by Darwin has had profound impact on biology. Most biologists are convinced that selection acting on advantageous mutations is the driving force of evolution. It was not until the late 1970s when molecular data became available, that the theory was challenged. In particular, Motoo Kimura found that the number of mutated substitutions in amino acid sequences of haemoglobin was too large to be explained by the theory of natural selection. Based on this discrepancy, he proposed the neutral theory, which states that most mutations at the molecular level in evolution are caused by random genetic drift rather than by natural selection (Kimura, 1968). In other words, the mutations involved are neither advantageous nor disadvantageous to the survival or reproduction of the individual. Around the same time, a similar theory was published by (King and Jukes, 1969). The two papers provoked much controversy and are still subject to strong debate (Kreitman, 1996).

Darwinism has been the dominating principle behind the implementation of evolutionary algorithms: the evolved entities contain no *explicit* neutral mutations. In this way, a mutation is either advantageous or disadvantageous. Selection acts upon them to propagate those that are advantageous.

But can neutral mutations benefit evolutionary search? To investigate this question, we have devised a methodology for systematic study of this subject (Yu and Miller, 2001). In particular, we measure the number of neutral mutations that occur in the evolved entities during evolutionary search. In this way, the impact of neutrality on search performance can be analyzed quantitatively. Using this approach, we have studied a Boolean function and four needle-in-haystack problems. The results show that there is a positive relationship between neutral mutations and the success rate: the larger the allowed quantity of neutral mutations the greater the possibility for the evolutionary search to find a solution.

The amount of neutral mutations is measured in the selection step, which evaluates both the fitness and the number of neutral mutations in the evolved entities. More precisely, an offspring solution is selected to replace the current winner only when it has a better fitness or it has the same fitness but its neutral mutations are within a specified range (the Hamming bound). One can envisage all solutions with the same fitness that satisfy the Hamming bound to be connected in a network (neutral network). The search process selects solutions in the network one after another in the manner of a neutral walk. We found that such a walk can lead to a solution with a better fitness if it satisfies the fitness improvement criterion. The criterion used is the ratio of adaptive and neutral mutations. The analysis shows that when this ratio for the neutral walks was close to the ratio for the fitness improvement, a high probability of success occurred.

Adaptive and neutral mutations play different roles in the evolutionary search: adaptive mutations exploit the accumulated beneficial mutations while neutral mutations provide an exploratory power by maintaining genetic diversity. Under the dynamics of the evolutionary process, neutral mutations may contribute to the fitness later. For an evolutionary search to be successful, it requires a balance between exploitation and exploration. The ratio between adaptive and neutral mutations is therefore an appropriate fitness improvement criterion in evolutionary search.

3 CARTESIAN GP VS. SIMPLE GA

In Cartesian Genetic Programming (CGP) (Miller and Thomson, 2000), a genotype encodes an indexed, feed-forward, acyclic graph. Unlike the fixed-length binary string representation in Simple Genetic Algorithms (SGA), a graph allows its nodes to be unconnected to any other nodes, hence has variable length. The nodes that are not involved in the connected path between the input and output of the genotype are inactive. They have no effect on the fitness of the genotype. For example, Figure 1(a) is a OneMax function with 6 nodes; each has 2 genes (one input link values and one function value). The function gene (in bold) takes value from the set `{0,1}`, which represent the function `add0` and `add1` respectively. The node output links are labeled from 1 to 6.

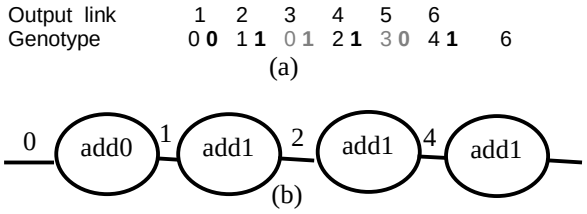

Figure 1: OneMax genotype (a) and encoded graph (b)

The encoded graph in 1(b) is obtained from the genotype by a decoding process that begins at the rightmost gene (6) and proceeds towards the left. The number 6 refers to the node (with genes 4 1). It is an `add1` node with input connected to the node with output link 4. The node with output link 5 does not appear in the graph (b), because it is not referred to by any active nodes. The same applies to the node with output link 3. These two inactive nodes are grayed in 1(a) to show this.

Mutations applied to inactive genes are neutral while on active genes they can be adaptive or neutral (see Section 3.1). This genotype representation, in which some genes are active and others inactive, allows neutral mutations to be measured by Hamming distance. This is described in Section 3.1.

3.1 NEUTRALITY MEASURED AS HAMMING DISTANCE

When a mutation operation generates a different genotype with the *same* fitness, it involves a number of neutral

mutations. Neutral mutations on active genes are the result of functional redundancy or introns, and hence provide *implicit* neutrality. In contrast, neutral mutations on inactive genes provide *explicit* neutrality. Regardless of the source of neutrality (implicit or explicit), the overall amount of neutral mutations between a parent-offspring genotype pair can be measured according to their Hamming distance. For example, the following two genotypes have Hamming distance 8.

Output link	1	2	3	4	5	6	
Genotype 1	0 1	0 0	1 1	2 0	3 0	5 1	6
Genotype 2	0 0	1 1	1 0	3 0	4 1	5 0	6

The number of active genes changes between the two genotypes is seven (the node with output link number four contributes two active genes changes because it was inactive in Genotype 1 but becomes active in Genotype 2). The number of inactive genes changes is two (corresponding to the node with output link number 2).

Using Hamming distance to measure neutral mutations has two advantages:

- It provides a quantitative measurement of neutrality in evolutionary search.
- It provides a flexible way to control neutrality in evolutionary algorithms.

4 RELATED WORK

Levenick introduced the idea of inserting introns (non-coding regions) in GA representation (Levenick, 1991). Unlike inactive genes in CGP which may become active (and vice versa) as the result of evolutionary dynamics, non-coding genes remain silent throughout the evolutionary process. He tested this coding scheme on a simple 5-exons (each has 6 genes) problem and reported that they improve search success rates. Although he proposed a couple of explanations for why introns improved success rates, no formal study was provided.

Later, Forrest and Mitchell used the same coding mechanism to test two Royal Road functions. They reported that no improvement is gained (in terms of the number of function evaluations takes to find the optimum) by including non-coding segments in their GA (Forrest & Mitchell, 1992).

Wu and Lindsay repeated the same experiments of Levenick and Forest & Mitchell. They reported the similar results as those reported by previous researchers (Wu and Lindsay, 1995). Additionally, they extended their experiments on the Royal Road problem with different fitness functions: one gave an exponential increase in fitness when a building block was found; the other gave a power law increase of fitness when a building block was found. They reported that non-coding segments impair the GA performance (in terms of the number of generation takes to find the optimum) in the first case but improve the performance in the second case.

Another set of experiments they conducted is concerned with increasing the level (the number of layers that a building block is broken down into) of the Royal Road problem. They reported that non-coding segments reduced the number of generation to find the optimum in this set of experiments. They also discussed hypotheses about the role of non-coding segments in GA, such as reducing the hitchhiking effect and stabilizing the discovered building blocks and preventing their loss.

Royal Road is a specifically designed problem to work with building block hypothesis. The solutions therefore consist of building blocks, which in turn are made of more than one gene. Our previous work on Boolean function and needle-in-haystack problems has different landscapes without building blocks. Similarly, the OneMax problem in this study does not have building blocks either. It might be interesting to test CGP on the Royal Road functions.

The linkage learning GA has a representation that also contains links (Harik and Goldberg, 1996). Similar to that of CGP, each gene has two elements: a value and a location (link). However, unlike that in CGP, genes with duplicated locations are deleted from the genotypes. Such duplications are normally the results of crossover and mutation. Instead of retaining them as inactive genes (which CGP does) to maintain diversity, these redundant genes are removed. As a result, no explicit neutrality is supported in the linkage learning GA.

One interesting application related to neutral mutations is on modeling the role of neutral and selective mutations in cancer (Maley and Forrest, 2000). Two dominant characteristics of cancer cells are their genetic instability and uncontrolled proliferation. The SEER report from the National Cancer Institute (Ries et al., 1998) estimates the lifetime probability of being diagnosed with cancer in the US is 45% for men and 38% for women. To match this estimate, they made a rough calculation that in general cancers require 3 selected mutations (that produce genetic instability) and these mutated clones would tend to spread to population of 10^7 cells (uncontrolled proliferation).

Using a 2-dimensional cellular automaton, they modeled the evolution of pre-cancerous cells. The results suggested that there must be at least 2 neutral mutations for the development of cancer.

5 ONEMAX SEARCH SPACE

With the presence of extra inactive genes in its genotypes, the search space of CGP is different from that of SGA. For example, this work solves OneMax with maximum fitness 25. With genotypes represented by 25 nodes (each contains a gene value of 1 or 0), the size of the SGA search space is 2^{25}.

A SGA genotype may have one of 26 possible fitness (0 to 25). A fitness value *i* means that *i* of the 25 genes have value 1 while others have value 0. The number of genotypes with fitness *i* is therefore:

$$\binom{25}{i} = \frac{25!}{i!(25-i)!}$$

In other words, although there is no explicit neutrality in the SGA representation, there is *implicit* neutrality (functional redundancy). Figure 2 gives the number of genotypes for the 26 different fitness groups. It shows that more than 95% of the search space are genotypes with fitness between 8 and 17. High fitness (> 20) genotypes are very few in comparisons. There is only one genotype with the optimum fitness of 25.

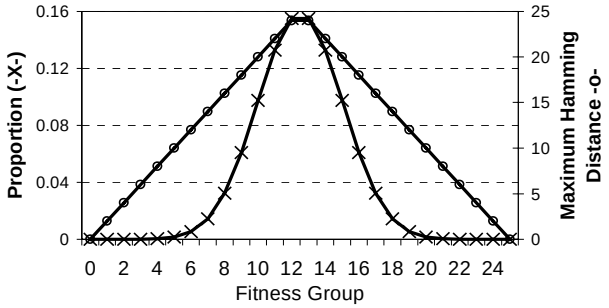

Figure 2: SGA OneMax Search Space.

Within each of the 26 fitness groups, we can measure neutrality between any two genotypes using their Hamming distance as described in Section 3.1. What we are interested in knowing is the maximum Hamming distance in each fitness group. This information quantifies the amount of *implicit neutrality* supported in the SGA genotype representation.

For each fitness group *i*, the maximum Hamming distance occurs for two genotypes whose 1-value genes have the least number of overlaps. This minimum overlap (O_{min}) value is dependent on the length of the genotype (25).

$$O_{min} = \begin{cases} 0, & i < 13 \\ 2i - 25, & \text{otherwise} \end{cases}$$

$$H_{max} = \begin{cases} 2i, & i < 13 \\ 25 - O_{min}, & \text{otherwise} \end{cases}$$

For example, the O_{min} for fitness group 1 is 0, and its maximum Hamming distance is 2.

```
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

The O_{min} for fitness 13 group is 1 and its maximum Hamming distance is 24.

```
1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1
```

Figure 2 shows that fitness 12 and 13 groups have the largest maximum Hamming distance (24). They also constitute the largest portions (15.4%) of the search space. In other words, the higher the maximum Hamming distance (implicit neutrality) in a fitness group, the larger the number of genotypes in the search space.

In contrast, the CGP representation supports not only *implicit* but also *explicit* neutrality, through extra inactive nodes. In this study, we increase the genotype length from 25 to 50 nodes. Moreover, each node contains two genes: one stores a link and the other stores a function (`add1` or `add0`). The link is the node number to where the current node is linked.

Additionally, each genotype has one input and one output links. Nodes that are not involved in the connected path between the input and output links are inactive. During the evolutionary search process, where both link and function values are evolved, an inactive node may become active. Similarly, an active node may become inactive.

Since the genotype represents a feed-forward graph, a node can only be linked to nodes before it. The link value of the first node, therefore, has only 1 possibility: a constant 0. The second node has 2 possible link values (a constant 0 and the output link of the first node) and so on. Moreover, since each function node value can be either `add0` or `add1`, the number of genotypes in the search space is:

$$(2*1) * (2*2) * (2*3) \ldots (2*50) = 2^{50} * 50!$$

These 25 extra nodes undoubtedly increase the number of genotypes in the search space, as they give more ways to construct a genotype with the same fitness. A genotype with fitness *i* is one which has *i* of its connected nodes with function values `add1`.

We used the Stirling number's (Graham, Knuth and Patashnik, 1994) of the first kind to calculate the number of genotypes of each fitness group in the search space. With a genotype length of 50 and *n* active nodes (i.e. *n* nodes are linked while 50 - *n* nodes are unlinked), the number of such kind of genotype is equal to $S(50,n)$.

For a genotype to have fitness *i*, the linked portion can be of *i* to 50 nodes. Moreover, *i* of the *n* linked nodes are of function `add1` while the 50-*i* unlinked nodes are of either `add0` or `add1`. The total number of genotypes with fitness *i* is given by:

$$\sum_{k=i}^{50} \binom{k}{i} 2^{(50-k)} S(50,k)$$

When *i* is small, only a small number of nodes have to be linked. The rest of them can be unlinked. Since unlinked nodes can have more different ways of being constructed, the number of possible genotypes is larger. Figure 3 shows that a genotype with fitness 3 constitutes about 29 percent of the search space. The number of genotypes decreases as the fitness value increases. Although the proportion of the search space with genotype fitness 25 is very small, there exist more the one such optimum, which is larger than that (1) in the SGA search space.

We also analyze the maximum Hamming distances in each of the 26 fitness groups. Since the genotype has 50

nodes, the minimum overlap for all fitness *i* (0 to 25) is 0. For example, a genotype with fitness 25 can be one which has its first 25 nodes connected and with `add1` while the other 25 disconnected nodes with `add0`. Another fitness 25 genotype can be the reverse of the previous one: the first 25 nodes are disconnected and have `add0` while the rest 25 nodes are connected and with `add1`. These two genotypes have 0 overlap (Hamming distance is 100). The same logic applies to all fitness groups; hence all have a maximum Hamming distance of 100. By supporting a larger amount of neutrality (in terms of Hamming distance) than an SGA (100 vs. 24), CGP search space contains more genotypes with the same fitness. We will evaluate the impact of this increased neutrality on evolutionary search through a series of experiments.

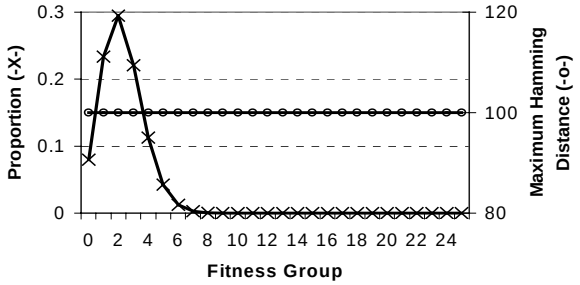

Figure 3: CGP OneMax Search Space.

6 EXPERIMENTAL SETUP

We conducted experiments using two different representations (SGA and CGP). Moreover, the CGP representation is combined with a selection scheme that considers the Hamming distance between parent and offspring. We first describe the evolutionary algorithm used in the experiments in Section 6.1. The control parameters for CGP and SGA representations are given in Section 6.2.

6.1 EVOLUTIONARY ALGORITHM

The algorithm is a form of 1+λ evolutionary strategy, where λ=4, i.e. one parent with 4 offspring (population size 5). The algorithm is as follows:

1. Randomly generate an initial population of 5 genotypes and select the one with the best fitness as the winner.
2. Carry out point-wise mutation on the winning parent to generate 4 offspring;
3. Construct a new generation with the winner and its offspring;
4. Select a winner from the current population using the following rules:
   - If any offspring has a better fitness, it becomes the winner.
   - Otherwise, an offspring with the same fitness is randomly selected. If the parent-offspring pair has a Hamming distance within the permitted range (see Section 6.2), the offspring becomes the winner.
   - Otherwise, the parent remains as the winner.
5. Go to step 2 unless the maximum number of generations reached or a OneMax solution is found.

6.2 CONTROL PARAMETERS

For SGA experiments, the genotype length is 25. The maximum fitness is 25 (where all genes have function value `add1`). During the selection step, Hamming distance was *not* checked. In other words, any offspring with the same fitness as that of the parent becomes the new winner. As analyzed in Section 5, SGA representation supports implicit neutrality with different Hamming distances ranging from 0 to 24. This is the amount of neutrality used during evolutionary search.

For CGP experiments, the genotype length is 100 (50 nodes). The maximum fitness is also 25. During the selection step, Hamming distance is checked. In particular, we used 11 different Hamming distances range from 0 to 100. When Hamming distance is less than 24, the amount of neutrality is less than that supported by the SGA representation. These setups allow us to assess the impact of different neutrality on the search performance.

	SGA	**CGP**
Genotype Length	25 genes (25 nodes)	100 genes (50 nodes)
Max Fitness	25	25
Functions Set	`add0` and `add1`	`add0` and `add1`
Hamming Distance	not checked	0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100
Max Gen	10,000	10,000
Pop Size	5	5
Runs	100	100
Mutation (# of genes)	1, 2, 3, 4, 5	1,2,3,4,5,6,7,8,9,10

Table 1: Control Parameters.

7 EXPERIMENTAL RESULTS

The CGP experimental results show that higher Hamming distances give higher average success rates (see Figure 4a). When Hamming distance 0 is used (no neutrality is permitted), the average success rate is 0% across all mutation rates. The success rate starts to increase consistently when Hamming distance 20 is used. This increase continues until Hamming distance 50 is reached. All Hamming distances beyond 50 have a similar success rate as that of Hamming distance 50.

Two-point mutation is most effective with this problem. Increasing mutation rate reduces the success rates when small Hamming distances are used. This suggests that mutation is destructive and high Hamming distance is able to absorb such destruction, hence maintain the search performance. However, when 6-point mutation is reached, the average success rates decrease constantly, regardless of Hamming distances.

The average final fitness has a very similar performance pattern to those of the average success rates (see Figure 4b). The only exception is that while Hamming distance 10 does not increase average success rate, it improves the average final fitness.

Section 5 shows that SGA representation gives an implicit neutrality level between 0 and 24, depending on the fitness group. One would anticipate that its performance would be between Hamming distance 20 and 50 in CGP. Indeed, its average final fitness values are within such ranges (see Figure 4b). However, its average success rates are not within this range. Most of the success rates are worse than expected (see Figure 4a). In other words, although most of the SGA runs reach expected fitness (23 and 24), the number of runs reaching the optimum of 25 is less than what were expected. This indicates the implicit neutrality provided by SGA for high fitness groups (2 for fitness 23 and 1 for fitness 24) is not sufficient to absorb the destructive mutations, hence unable to improve the last few bits to reach the optimum.

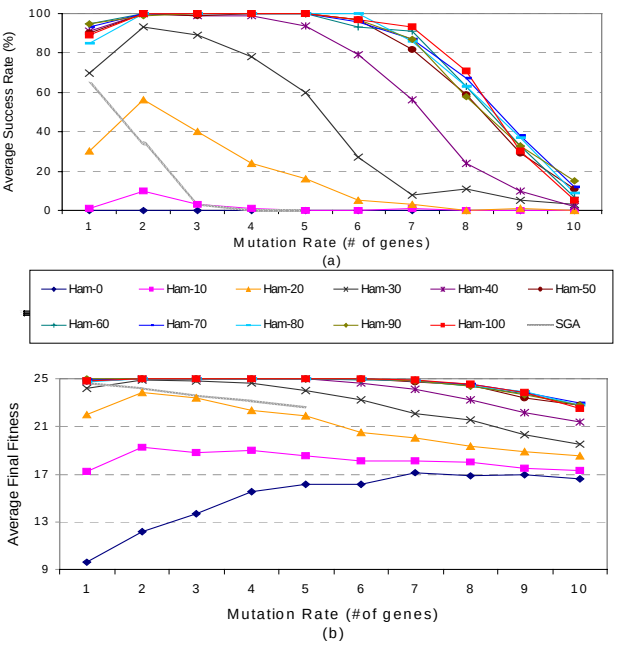

Figure 4: CGP & SGA Experimental Results.

The following conclusions can be drawn from the CGP experimental results:

- Mutation is destructive when the population becomes fit. Neutrality provides buffers to protect these destructive mutations and gives better final fitness and higher success rates.
- Hamming distance 50 is the equilibrium point, which gives the exploration/exploitation balance corresponding to the maximum possible performance. Increasing neutrality beyond this point has little effect on the performance.

8 THEORETICAL ANALYSIS

While experimental results show a positive relationship between the size of Hamming distance and the success rate in this unimodal landscape, we intend to model the search process mathematically to understand this correlation. In particular, the search process is concerned with the ratios of active and inactive gene changes (see Section 2). Although the work is not conclusive, we report here the interim results.

Each CGP genotype has two parts: inactive and active. Moreover, mutations on inactive and active genes have different effects. We consider these below:

- Inactive gene (function or link gene): only the mutated gene is changed. No other side effect occurs.
- Active function gene: when the number of `add1` flipped to `add0` is greater the number of `add0` flipped to `add1`, the fitness decreases. If the two numbers are the same, the fitness remains the same. Otherwise, the fitness increases. Besides the possible fitness changes, no other side effect takes place.
- Active link gene: there are complicated side effects since the mutated link can cause inactive genes to become active and vice versa. Both of these two kinds of changes are counted toward active gene changes (see Section 4). An active link gene change can increase, decrease or have no effect to the fitness.

When the current winner has fitness *i*, the number of active and inactive gene changed can be calculated in the following way.

1. For a genotype of fitness *i* with *k* linked nodes, the probability that a single-point mutation hits an inactive gene, an active gene, an `add0` gene, an `add1` gene and an active link gene is as follows:

$$p(\text{active}) = \frac{2k}{100} \quad (1)$$

$$p(\text{inactive}) = \frac{2(50-k)}{100} \quad (2)$$

$$p(\text{add1}) = p(\text{add1Link}) = \frac{i}{100} \quad (3)$$

$$p(\text{add0}) = p(\text{add0Link}) = \frac{k-i}{100} \quad (4)$$

$$p(\text{activeLink}) = \frac{k}{100} \quad (5)$$

2. When there are M mutations (assume each one is independent to each other), some of them will be on active and others on inactive genes. For example, a two-point mutation can be of one of the 4 cases:

Case	No. of inactive gene changes
(active,active)	$p(active)^2$ x $p(inactive)^0$ x 0
(active,inactive)	$p(active)^1$ x $p(inactive)^1$ x 1
(inactive,active)	$p(inactive)^1$ x $p(active)^1$ x 1
(inactive,inactive)	$p(inactive)^2$ x $p(active)^0$ x 2

Table 2: 4 possible cases for two-point mutation.

The number of *inactive* gene changes under a M-point mutation on genotype with *k* linked nodes is therefore:

$$Inact(M,k) = \sum_{m=0}^{M} p(inactive)^m p(active)^{M-m} \binom{M}{m} m \quad (6)$$

2. For a fitness *i* group, each genotype may have a different number of linked nodes. The number of genotypes that have *k* nodes linked is:

$$Linked(k,i) = \binom{k}{i} 2^{(50-k)} S(50,k) \quad (7)$$

3. The total number of genotype with fitness *i* is the sum of the genotype with different number of linked (active) nodes from *i* to 50.

$$Total(i) = \sum_{k=i}^{50} \binom{k}{i} 2^{(50-k)} S(50,k) \quad (8)$$

4. Hence, for a fitness *i* group, the total number of *inactive gene* changes under a M-point mutation is:

$$Inactive(M,i) = \sum_{k=i}^{50} Inact(M,k) \frac{Linked(k,i)}{Total(i)} \quad (9)$$

5. Similarly, we can analyze the number *of active function gene* changes that don't affect the fitness of a genotype. To maintain the same fitness, the number of active `add1` flipped to `add0` has to be the same as the number of active `add0` flipped to `add1`. This means the minimum number of mutation points is 2 and the minimum number of linked node has to be *i*+1.Table 3 shows 7 possible cases of a 3-point mutation that maintain the same fitness:

Case	No. of active fun gene changes
(add0,add1,other)	p(add0) x p(add1)x p(other)x 2
(add0,other,add1)	p(add0) x p(other)x p(add1)x 2
(add1,other,add0)	p(add1) x p(other)x p(add0)x 2
(add1,add0,other)	p(add1) x p(add0)x p(other)x 2
(other,add1,add0)	p(other) x p(add1)x p(add0)x 2
(other,add0,add1)	p(other) x p(add0)x p(add1)x 2
(other,other,other)	$p(other)^3$ x 0

Table 3: 7 possible cases of a 3-point mutation.

Note: p(other)= p(inactive)+p(activeLink)

The total number of *active function gene* changes (that maintain the same fitness) under a M-point mutation with *k* linked nodes can be shown to be:

$$ActFunNeu(M,k) = \sum_{m=0,2}^{M} (p(add0)p(add1))^{m/2} \binom{m}{m/2} p(others)^{M-m} \binom{M}{M-m} m \quad (10)$$

6. For a fitness *i* group, the total number of *active function gene* changes under a M-point mutation that maintains the same fitness *i* is:

$$ActiveFunNeu(M,i) = \sum_{k=i+1}^{50} ActFunNeu(M,k) \frac{Linked(k,i)}{Total(i)} \quad (11)$$

7. To increase the fitness, the number of active `add1` flipped to `add0` has to be less than the number of active `add0` flipped to `add1`. Following the previous logic, the number of `add0`, `add1` and other gene changes are:

$$add1Imp = p(add1)^{Min(M-m,\,m-1)} \binom{M}{Min(M-m,\,m-1)} \quad (12)$$

$$othersImp = p(others)^{M-m-Min(M-m,\,m-1)} \binom{M}{M-m-Min(M-m,\,m-1)} \quad (13)$$

(14)

The number of *active function gene* changes for a M-point mutation that improve the fitness *i* is:

$$ActFunImp(M,k) = \sum_{m=0}^{M} add0Imp \times add1Imp \times othersImp \times m \quad (15)$$

8. For a fitness *i* group, the total number of *active function gene* changes under a M-point mutation that improves fitness *i* is:

$$ActiveFunImp(M,i) = \sum_{k=i}^{50} ActFunImp(M,k) \frac{Linked(k,i)}{Total(i)} \quad (16)$$

9. A single mutation on an *active link* gene can make more than 1 active gene changes. The following example shows that a single mutation on the link gene on node with output link 5 has produced 9 active gene changes.

Output link	1	2	3	4	5	6
Genotype 1	0 1	0 0	1 1	2 0	**3** 0	5 1 6
Genotype 2	0 1	0 0	1 1	2 0	**4** 0	5 1 6

This complexity makes it hard to formulate a mathematical equation that give the number of active gene changes under active link mutations. We are currently working on this last part of the whole puzzle.

Based on the above study, we have calculated the number of active and inactive gene changes under a M-point mutation for the fitness 10 group. These are shown in Figure 5. It shows clearly that the number of all 3 types of gene changes increases linearly as mutation points increase.

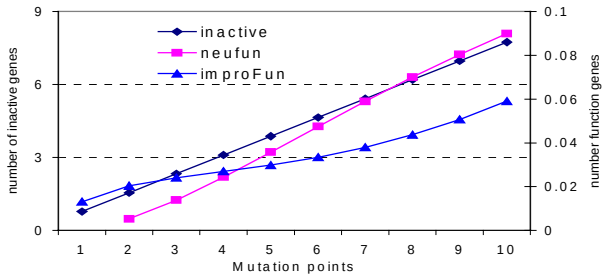

Figure 5: Number of the 3 types of gene changes.

The search space on which our study is based is essentially the search space of Hamming distance 100 implementation. Figure 6 gives the active/inactive gene changes ratios for neutral walk and fitness improvement. Note that the ratios are based on gene changes in a single step (1 generation). For multiple generations, we anticipate a Markov chain model might be needed to give the complete picture of the search process.

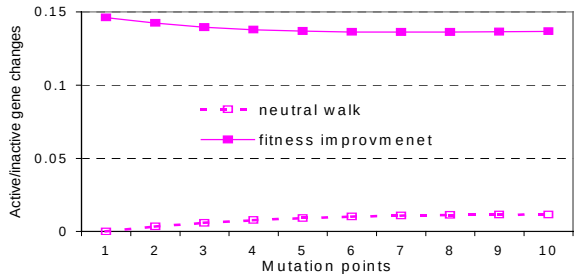

Figure 6: Hamming Distance 100 Search Process.

9 CONCLUSIONS

Neutrality in evolutionary algorithms in general, and in GAs in particular, is not fully understood. This work on the application of our a neutrality-enabled evolutionary system to the OneMax problem has made the following contributions:

- It provides a quantitative analysis of neutrality supported in the SGA and in the CGP search spaces for this problem.
- It shows that for this unimodal landscape, neutrality is advantageous to evolutionary search because it provides buffers to absorb destructive mutations (occurred when the population is fit). As a result, higher success rates can be obtained when sufficient neutrality is provided during the search.
- It presents a framework for the theoretical study of search behavior of the system. By modeling the active and inactive gene changes, the role of neutrality in evolutionary search can be better understood.

We continue the work on the theoretical analysis of the system, especially the effects of mutations on active link genes. We are also investigating scalability of neutrality in evolutionary search for Boolean function landscapes.

10 REFERENCES

Ackley, D. H. (1987). *A Connectionist Machine for Genetic Hillclimbing*. Kluwer Academic Publishers.

Bäck, T. (1992). The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm. *Proceedings of Parallel Problem Solving from Nature II*, Elsevier, Amsterdam, Pages 85-94.

Forrest, S. and Mitchell, M. (1992). Relative building-block fitness and the building-block hypothesis. In *Foundations of Genetic Algorithms 2*, p. 109-126.

Giguere, P. and Goldberg, D. E. (1998). Population size for optimum sampling with genetic algorithms: a case study of the onemax problem. *Proceedings of the Third Annual Genetic Programming Conference*. pp. 496-503.

Graham R. L, Knuth D., and Patashnik O. (1994). Concrete Mathematics: a Foundation for Computer Science. Addison Wesley.

Harik, G. R., and Goldberg, D. E. (1996). Learning linkage. *Foundations of Genetic Algorithms 4*, p 247-262.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor. MI: University of Michigan Press.

Kimura, M. (1983). *The Neutral Theory of Molecular Evolution*. Cambridge Univ. Press.

Kreitman, M. (1996). The neutral theory is dead, Long live the neutral theory. *BioEssays*, Vol. 18. 678-682.

King, J. L. and Jukes, T. H. (1996). Non-Darwinian evolution. *Science* Vol. 164, Pages 788-798.

Levenick, J. R. (1991). Inserting introns improves genetic algorithm success rate: Taking a cue from biology. *Proceedings of the Fourth International Conference on Genetic Algorithms*. Pages 123-127.

Miller, J. F. and Thomson, P (2000). Cartesian genetic programming. *Proceedings of the Third European Conference on Genetic Programming.* Pages 121-132.

Maley, C. C. and Forrest, S. (2000). Modeling the role of neutral and selective mutations in cancer. *Artificial Life VII: Proceedings of the Seventh International Conference*, MIT Press, Pages 395-404.

Ries, L. A. G., Kosary, C. L., Hankey, B. F., Miller, B. A., and Edwards, B. K., editors (1998*). SEER Cancer Statistics Review*, 1973-1995. National Cancer Institute.

Wu, A. and Lindsay, R. K. (1995). Empirical studies of the genetic algorithm with non-coding segments. *Evolutionary Computation*, 3:2, 121-147.

Yu, T. and Miller, J. (2001). Neutrality and the evolovability of Boolean function landscape. In *Proceedings of the Fourth European Conference on Genetic Programming*. Pages 204-217.

Yu, T. and Miller, J. (2002). Finding needles in haystacks is not hard with neutrality. In *Proceedings of the Fifth European Conference on Genetic Programming*. Pages 13-25.