Neutrality and the Evolvability of Boolean Function Landscape

Tina Yu[1] and Julian Miller[2]

[1]Chevron Information Technology Company, San Ramon CA 94583, U.S.A.
tiyu@chevron.com, http://www.addr.com/~tinayu

[2]School of Computer Science, University of Birmingham, Birmingham B15 2TT, U.K.
j.miller@cs.bham.ac.uk, http://www.cs.bham.ac.uk/~jfm

Abstract. This work is a study of neutrality in the context of Evolutionary Computation systems. In particular, we introduce the use of explicit neutrality with an integer string coding scheme to allow neutrality to be measured during evolution. We tested this method on a Boolean benchmark problem. The experimental results indicate that there is a positive relationship between neutrality and evolvability: *neutrality improves evolvability*. We also identify four characteristics of adaptive/neutral mutations that are associated with high evolvability. They may be the ingredients in designing effective Evolutionary Computation systems for the Boolean class problem.

1 Introduction

The Darwinian theory of evolution is based on two principles: *selection* and *mutation*. It states that through the process of natural selection, organisms become progressively adapted to their environments by accumulating beneficial mutations. During the study of population genetics, Sewall Wright observed that "genetic drift" (the random change in the frequency of alleles in a population that are not related to selection) plays an important role in the process of evolution [17]. Although it did not deny the role of selection in evolution, Motoo Kimura's neutral theory argued that the majority of evolutionary changes at the molecular level are the results of random fixation of selectively neutral mutations rather than the results of Darwinian selection acting on advantageous mutations. In other words, the mutations involved are neither advantageous nor disadvantageous to the survival and reproduction of individuals. Such random genetic drift, as the theory claims, should be considered as an important explanatory factor of evolution [7,9]. This radical view has led to a controversy (often called "neutralist-selectionist controversy") which still is a matter for strong debate in Evolutionary Biology. Nevertheless, new data supporting neutral theory continues to be reported [8].

Evolutionary Computation (EC) systems model the process of natural evolution for problem solving. With strong Darwinian influences, most EC systems adopt a selectionist's point of view to model evolution process, i.e. no explicit neutral mutations are coded in the evolved entities. One interesting EC paradigm is Genetic Programming (GP) [10] where neutral mutations are implicitly embedded in its evolved computer programs through *functional redundancy* and *introns* (see Section 2). There have been studies reported on these implicit neutral mutations [1]. In this work, we study neutrality in general through *explicit* neutral mutations. Moreover, the relationship between neutrality and evolvability of a GP system is analyzed.

The concept of evolvability has interested many researchers who want to understand the characteristics of evolvable systems so that they can incorporate these ingredients in their EC systems to evolve complex problem solutions. According to Altenberg, evolvability is the ability of a population to produce variants fitter than any yet existing. He believes that to achieve high evolvability, "conservation of fitness score" during evolution is very important. This "conservedness", however, is difficult to obtain when the population has become fit and mutation is disruptive in transmitting the fitness of an individual to its offspring. The implicit neutral mutations in the GP program representation provide a buffer against disruptive mutations; this enables more fitter offspring to be generated [1]. The work of Hinton and Nowlan showed that lifetime learning of individuals can increase evolvability. To solve a "needle in a haystack" problem (Only the genotype with the correct combinations is rewarded with the perfect fitness. Any other genotypes are rewarded with fitness value 0), they put a redundant gene "?" in the genotype representation. The experimental results show that evolutionary search was able to learn to use the new representation and to generate fitter solution [5].

The goal of this paper is to elucidate quantitatively how neutrality improves the evolutionary search process for a Boolean benchmark problem (even-3-parity). The work extends earlier findings on the advantages of neutrality for digital circuit evolution [19]. Evolving Boolean functions is a challenging endeavor and is likely to be important in the Evolvable Hardware research community. Moreover, we believe that understanding the advantages and role of neutrality on this class of problem will be of interest to the wider research community.

The paper is organized as the following: Section 2 explains neutrality in the context of EC systems. Section 3 describes the Cartesian GP system that we used to conduct our experiments. In Section 4, we describe the implementations of our experiments. The results are presented in Section 5. In Section 6, the interaction between neutrality and evolvability in the fitness landscape is analyzed. Section 7 summaries related work and Section 8 gives our conclusions and future work.

2 Neutrality: Implicit Versus Explicit

The standard GP program representation provides two forms of implicit neutral mutations: functional redundancy and introns. Functional redundancy refers to many different programs (genotypes) representing the same function (phenotype). For example, the following 3 genotypes represent the same function `xor`:

G1: nor (and x_1 x_2) (nor x_1 x_2)
G2: nor (nand (nand x_1 x_2) (or x_1 x_2)) (nor x_1 x_2)
G3: nor (and x_1 x_2) (nor (nor x_1 x_2) (nand x_1 x_2))

Genetic transformation from one genotype to another, e.g. `G1` to `G2`, has a neutral effect on the program's behavior. Functional redundancy therefore provides neutrality during programs evolution process.

Introns refer to the code that are part of a program but are semantically redundant to the program's behavior. For example, the function `or` with input `False` are introns in the following genotype:

```
G4: or False (nor (and x₁ x₂) (nor x₁ x₂))
```

Genetic transformation by removing introns from a genotype, e.g. `G4` to `G1`, has neutral effect on the program's behavior. Introns therefore also provide neutrality in the programs evolution process.

Functional redundancy and introns can emerge within an evolving genetic program. They are not easy to identify or control during the program evolution process. We therefore propose including extra code in the genotype to provide *explicit neutrality*. In this approach, a genotype can have part of its genes active and the others inactive. For example, the following 2 genotypes have extra code that is not active (in gray), yet still represent the `xor` function:

```
G5: nor (and x₁ x₂) (nor x₁ x₂) (nor x₁ x₂)
G6: nor (and x₁ x2) (and x₁ x₂) (nor x₁ x₂)
```

Genetic changes on inactive genes that transform one genotype to another, e.g. `G5` to `G6`, have a neutral effect on the program's behavior. Genotypes with inactive genes, therefore provide neutrality during the program evolution process.

The advantage of explicitly encoding neutral mutations in the genotypes is that it provides a way to measure neutrality in the evolutionary process. Using an integer string graph representation (see Section 3), neutrality between two genotypes with the same fitness can be measured by their hamming distance (see Section 4.3). Moreover, functional redundancy and introns can be measured using the same method (see Section 4.3). By studying explicit neutral mutations, we hope to understand neutrality in general and to be able to evaluate its impact on the evolvability of a GP system.

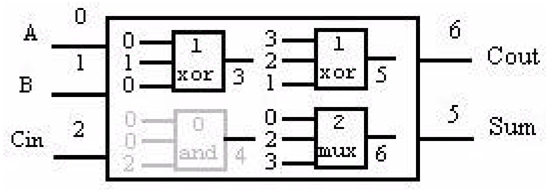
2.1 Neutral Versus Adaptive Mutation

Mutation on a genotype that has part of its genes active and others inactive produces different effects. Mutation on active genes is adaptive because it exploits the accumulated beneficial mutations. In contrast, mutation on inactive genes has neutral effect on a genotype's fitness, yet it provides an exploratory power by maintaining genetic diversity. Under the dynamics of the evolutionary process, active genes may become inactive while inactive genes may become active. Consequently, mutations on such kind of genotype representation provide two roles in the evolutionary search: exploitation and exploration. Yet, what is the best balance between neutral and adaptive mutation that would enable evolution to generate fitter offspring? We investigate this question using the Cartesian GP system on a Boolean function problem.

3 Cartesian Genetic Programming

Cartesian Genetic Programming (CGP) was originally formulated as an effective method for evolving digital electronic circuits, particularly Field Programmable Gate

Arrays (FPGA) [11,12,13]. In CGP, a program is represented as an indexed, directed, acyclic graph, which can be mapped to the physical structure of a FPGA. For example, a one-bit adder with carry is represented in Figure 1. This gate array consists of 4 logic gates (2 `xor`, 1 `and` and 1 `mux`). `A` and `B` and $\mathtt{C_{in}}$ represent the primary inputs. $\mathtt{C_{out}}$ and `Sum` are the output bits of the adder. The upper right `xor` gate (output link 5) has input connections 3,2,1. This means that the first input is connected to the output of the upper left `xor` gate. The second input is connected to the primary input $\mathtt{C_{in}}$, and the third input is connected to primary input `B`. The output $\mathtt{C_{out}}$ is connected to the output of the `mux` gate (lower right). This gate array is represented as a graph using an integer string in Figure 1(b). A gate array can have any number of rows and columns. Moreover, each gate can be connected to any other gates. The indexed, directed, acyclic graph is therefore a more natural representation than parse tree for digital electronic circuits.



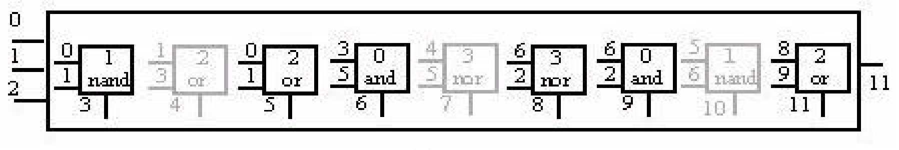
(a)

0 1 0 **1** 0 0 2 **0** 3 2 1 **1** 0 2 3 **2** 6 5

(b)

Figure 1 A one-bit adder with carry: (a) gate array representation (b) graph representation.

Unlike the parse tree representation, a graph allows its nodes to be unconnected to any other nodes. Those nodes that are not connected to the graph that links to the gate array's input and output nodes are inactive and have no effect on the behavior of the program. For example, Figure 2 (b) is an even-3 parity function (described in Section 4.1) represented as a graph with 3 inactive nodes:



(a)

0 1 **1** 1 3 **2** 0 1 **2** 3 5 **0** 4 5 **3** 6 2 **3** 6 2 **0** 5 6 **1** 8 9 **2**

(b)

Figure 2 An even-3-parity function: (a) gate array representation (b) graph representation.

The alleles in bold take value from the set {`0`, `1`, `2`, `3`}, which represent the two-arity Boolean functions `and`, `nand`, `or`, `nor` respectively. The program inputs are denoted by labels 0, 1, 2. The gate outputs are labeled from 3 to 11. There are 3 inactive nodes in this genotype, grayed in the graph representation.

4 Experiments

4.1 Even-3-Parity Problem

The problem studied in this paper is the even-3-parity Boolean function. This function takes 3 Boolean inputs and returns `True` only if an even number of inputs are `True`. The function set chosen was {`and`, `nand`, `or`, `nor`} and the terminal set was {x_0, x_1, x_2} which represent the three inputs to the program. There are 2^3 different possible inputs combination, hence 8 test cases. The fitness of a program is the number of the correct outputs for the 8 test cases, i.e. a program can have fitness between 0 and 8.

4.2 Evolutionary Algorithm

The experiments were conducted using a simplified (1+4) Evolution Strategy [18], i.e. one parent with 4 offspring (population size 5). The algorithm is described as follows:

1. Generate initial population of 5 programs randomly;
2. Evaluate fitness of the population;
3. Select the best of the 5 in the population as the winner;
4. Carry out point-wise mutation on the winning parent to generate 4 offspring;
5. Construct a new generation with the winner and its 4 offspring;
6. Select a winner from the current population using the following rules:
   - If any of the offspring has better fitness than the parent, it becomes the winner.
   - Otherwise, an offspring with the same fitness as the parent is randomly selected. If the parent-offspring genotype pair has hamming distances within the permitted range (see Section 4.3), the offspring becomes the winner.
   - Otherwise, the current parent remains as the winner.
7. Go to step 4 unless the maximum number of generations has reached.

For even-3-parity, each node has 3 genes (two inputs link values and one Boolean function value). To obtain statistically meaningful results, we made 100 runs on different mutation rate with different neutrality size (explained in the following section). Moreover, each run continues until the maximum number of generation is reached. Table 1 summarizes the parameters used to conduct the experiments.

Table 1: Summary of Parameters

Genotype Length	Mutation Rate (%) on Genotype	Max Generation
100 nodes (300 genes)	1,2,4,6,8,10,12,14,16,18,20	10,000

4.3 Neutrality Measured with Hamming Distance

When a mutation operation generates a different genotype with the same fitness during step 4 in Section 4.2, it involves a number of neutral mutations. Neutral mutations on active genes are the results of functional redundancy or introns, hence represent implicit neutrality. In contrast, neutral mutations on inactive genes represent explicit neutrality. Regardless the source of neutrality (implicit or explicit), the overall amount of neutral mutations between the parent-offspring genotypes pair can be measured according to their hamming distance. For example, the following two genotypes have hamming distances 9. The number of active genes changes between the two genotypes is 8 (the node with output link number 6 contributes 3 active genes changes because it was inactive in Genotype 1 but becomes active in Genotype 2). The number of inactive genes changes is 2 (corresponding to the node with output link number 4).

```
Output link       3      4      5     6       7
Genotype 1      2 1 3  0 1 2  1 3 0  4 3 2  1 5 9
Genotype 2      2 0 2  0 3 3  1 3 2  5 0 2  6 5 2
```

If neutrality is not permitted, evolution has to constantly provide fitness improvement, i.e. only a *better* offspring is accepted to replace current parent as winner. Hamming distance 0 indicates that neither implicit nor explicit neutrality is allowed to be present during evolution.

When the hamming distance range is a non-0 value, neutrality is allowed to be present, i.e. evolution can proceed in the presence of genetic drift where no fitness improvement occurs. The larger the permitted hamming distance range is, the larger the amount of neutral mutations is allowed during evolution. When the range of hamming distance permitted is the same as the length of the genotype (300 in this study), evolution can freely undergo unrestricted genetic drift until an improved offspring is obtained. In this work, we study the interaction between neutrality and evolution by experimenting with 6 different hamming distance thresholds (0, 50, 150, 200, 250, 300). The results are reported in the next section.

5 Results

The experimental results show that a larger amount of neutrality is better on this problem, regardless of the mutation rates. Figure 3 summarizes the average population fitness of 4 different mutation rates under the 6 different hamming distance implementations. These runs have initial average fitness between 3.96 and 3.99. However, for the purpose of contrasting the performance under different hamming distances implementation, only the fitness range that distinguishes their performance is plotted. As shown, hamming distance 250 and 300 implementations have very similar performance and generate better average fitness than the others. In other words, by supporting more freedom of genetic drift, evolution can generate fitter offspring, i.e. *neutrality improves evolvability*. We will analyze this interesting result in the following section.

6 Analysis and Discussion

There are 256 Boolean functions with 3 Boolean inputs and 1 Boolean output (see [10] page 216 for the listing). The 256 functions can handle the 8 test cases differently and are assigned different fitness accordingly. Figure 4 shows the fitness distribution of the 256 functions in the search space. Since there are many more functions with fitness 4, random search is more likely to generate a function with fitness 4. This explains why the average fitness of initial population is between 3.96 and 3.99 in the experiments.

The elitism selection mechanism used in our experiments keeps the best-so-far individual as the current winner. Moreover, the current winner can only be replaced by another genotype with *equal* or *better* fitness. When a genotype is replaced by a different genotype with the same fitness, evolutionary search is undergoing neutral walk in the search space. When a genotype with improved fitness is found, the search jumps into another partition of the search space and continues the process of neutral walk and the search of fitness improvement. In other words, evolutionary search is a progressive process that travels from one subspace with lower fitness functions to another subspace with higher fitness functions. We therefore analyze the evolutionary search within each of the subspaces, i.e. subspace of genotypes with fitness 5 and 6, 6 and 7, 7 and 8. We skip the subspace of genotypes with fitness 4 and 5 because it is very easy for evolutionary search to find a genotype with fitness 5. Also, we only use data of runs that have sequential fitness improvement, e.g. 4 to 5 to 6. This is because runs that had winning genotypes whose fitness are improved in a non-sequential manner (e.g. 4 to 6 to 7 to 8) were quite rare and they complicate the analysis of the search characteristics.

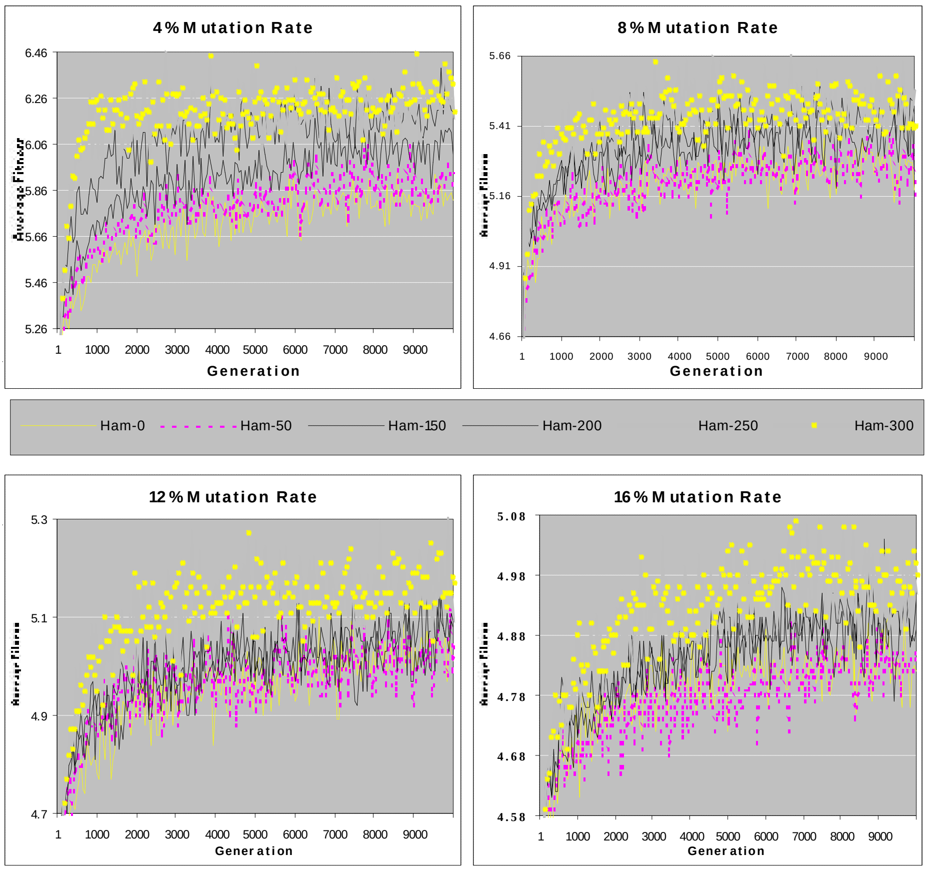


Figure 3 Average Population Fitness with Different Hamming Distances.

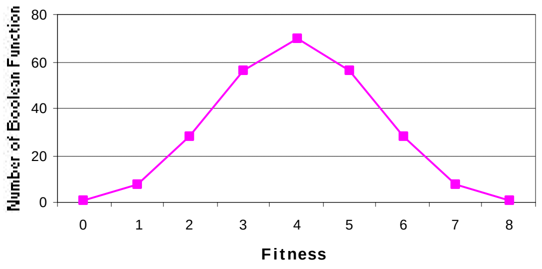


Figure 4 Fitness Distribution of Even-3-Parity Search Space.

6.1 Fitness 5 and 6 Search Space

With the presence of neutrality (hamming distance non-0 implementations), evolutionary search consists of two stages: neutral walk (the fitness of current winner remains 5) and the fitness improvement step (the fitness of current winner becomes 6). A neutral walk is a genetic drift process where both active and inactive genes undergo changes, although these changes do not produce fitness improvement. In Section 2.1, we have argued that the effect of active gene changes is exploitation while the effect of inactive gene changes is exploration. If the neutral walk can maintain the balance between exploitation and exploration, it is reasonable to believe that those gene changes will eventually lead to the fitness improvement step. In other words, for a genotype of fitness 5 to transform to become a genotype of fitness 6, adaptive *and* neutral mutations are required. The more the neutral walk can satisfy these mutations, the higher the probability of a genotype to reach the fitness improvement step.We test this hypothesis by investigating the relationship between the probability of success and the ratio of exploitation/exploration during the evolutionary search. The probability of success is the percentage of the genotypes with fitness 5 that successfully reach the fitness improvement step and are transformed into a fitness 6 genotype. The ratio of exploitation/exploration is defined as the number of active gene changes versus the number of inactive gene changes. We consider two such ratios in this study: one defined on the neutral walk and the other calculated from fitness improvement to the next fitness improvement. This information is calculated for various mutation rates. Figure 5 gives the results.

Figure 5(a) shows that this search space is very easy for the evolutionary algorithm to make the fitness improvement with most of the mutation rates and hamming distance thresholds. There are a few exceptions: hamming distance 0 with mutation rate 1%, 2%; hamming distance 50 with mutation rate 1%, 2%. (We ignore that of 99% success rate because they are too close to 100%). The hamming distance 0 implementation allows no neutral walk, i.e. a mutation operation has to transform a genotype of fitness 5 into a genotype of fitness 6 in one step. With low mutation rate, this can be quite difficult to achieve, hence the success rate is low. The same argument can be applied to the hamming distance 50 implementation.

An interesting question to ask is “why do other combinations of hamming distance and mutation rate generate high success rate?” We answer this question by analyzing the exploitation/exploration ratio during the neutral walk and at the fitness improvement step. Figure 5(b) shows that those runs with 100% success rates have the two ratios quite close to each other. For example, hamming distance 300 with mutation rate 10 generates an exploitation/exploration ratio of 0.43 at the fitness improvement step, i.e. for a genotype of fitness 5 to transform to become a genotype of fitness 6, the number of active genes changes versus the number of inactive changes is 0.43. The ratio provided by neutral walk is 0.40, which is very close to that is required for the successful transformation. This suggests that neutral walk has led the evolutionary search toward the fitness improvement step, hence enabled the genotype transformation to take place. We also examine the two combinations that have lower successful rates (hamming distance 50 with mutation rate 1% and 2%). In both cases, the two ratios are significantly different from each other. This suggests that neutral walk did not provide satisfactory adaptive/neutral mutations; hence evolutionary search fails to reach the fitness improvement step. These results endorse our hypothesis.

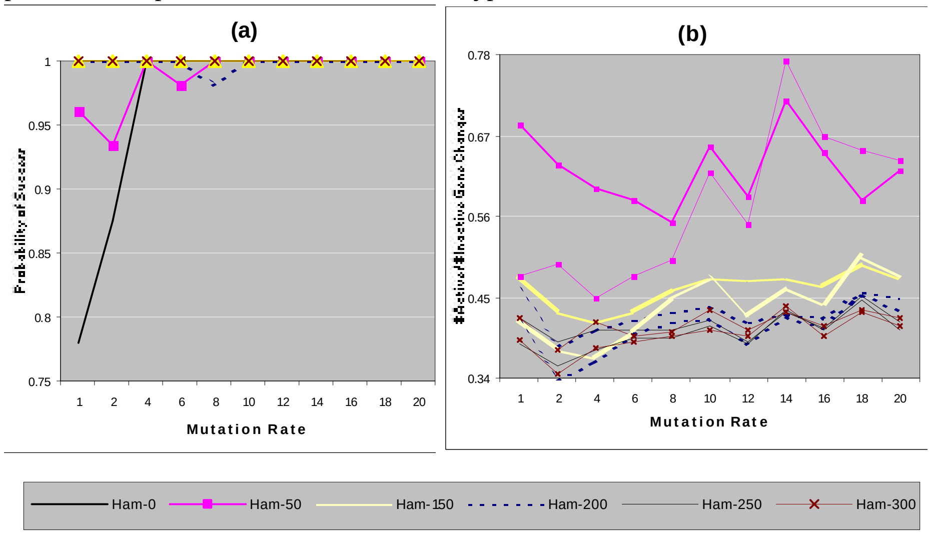


Figure 5 Fitness 5 and 6 search space.

6.2 Fitness 6 to 7 Search Space

The experimental results indicate that this search space is harder than the fitness 5 and 6 search space for most of the hamming distance implementations. Figure 6 (a) shows that only hamming distance 300 and 250 implementations still retain the 100% success rate across all mutation rates. The hamming distance 200 implementation only achieves 100% success for 6 of the 11 mutation rates. The success rate for hamming distance 150 implementation is about 95% across all mutation rates. In a similar way to that seen in the fitness 5 and 6 search space, hamming distance 50 and 0 implementations have better success rate when higher mutation rates are used. The lower the mutation rate, the lower the success rate. Once again, the exploitation/exploration ratios for neutral walk and for the fitness improvement step are very close to each other under the hamming distance 300 implementation. Across all mutation rates, the difference between the two ratios is about 0.009. A similar ratio difference is also exhibited in the hamming distance 250 implementation. These results, therefore, also support our hypothesis that appropriate adaptive/neutral mutations during neutral walk can lead to successful fitness improvement. The ratio difference with the hamming distance 200 implementation is a little bit larger (0.02). Nevertheless, the success rate is still reasonably high: 6 out of 11 mutation rates have 100% success rates. When hamming distance 150 is used, however, the ratios difference increases to about 0.04. This gap corresponds to a lower success rate (95%) on all mutation rates.

The results of the 4 hamming distance implementations discussed demonstrate some interesting patterns. First, the two target ratios are reasonably close to each other. Second, these ratio values are all within a small range between 0.3 and 0.45. Third, the ratio value increases as the mutation rate increases. This means when a higher mutation rate is applied, more active genes are changed, i.e. more exploitation. Finally, the ratio values are increased in a very smooth manner, which suggests the landscape under these implementations might be smooth. These 4 characteristics are also present in the fitness 5 and 6 search space.

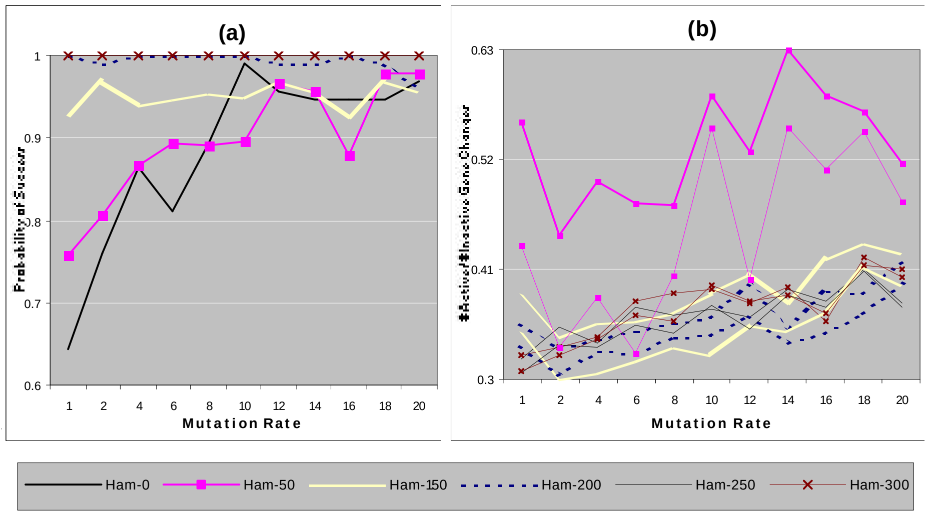


Figure 6 Fitness 6 and 7 Search Space.

In contrast, hamming distance 50 implementation has none of the features described. The two target ratios are distant from each other on most of the mutation rates. The two exceptions are mutation rate 18% ad 20%, where the ratio differences are 0.02 and 0.037 respectively. The experimental results show that their success rates are better than the others, which agrees with our hypothesis. However, a smaller ratio difference does not always correspond to higher success rate in this implementation. For example, mutation rate 10% has lower ratio difference than that of mutation rate 12% (0.03 vs. 0.127). Yet, it has a lower success rate (89% vs. 96%). We can not explain this phenomenon. However, we do suspect that the landscape under the hamming distance 50 implementation is rugged (based on the shape of the ratios curve). This may be a factor in the unpredictability of the results.

6.3 Fitness 7 to 8 Search Space

The experimental results indicate that this search space is the hardest one for all hamming distance implementations. Figure 7(a) shows that only hamming distance 250 and 300 were able to achieve 100% success rate on some of the mutation rates. Moreover, all implementations have success rates that are worse than those in the other two search spaces. As before, higher hamming distance values generate higher success rate, i.e. *neutrality improves success rates*. However, unlike the other two search spaces, success rate decreases when higher mutation rate is applied (except the hamming distance 50 implementation). This indicates that mutation is destructive in transmitting the fitness of an individual to its offspring in this search space. This result is consistent with that reported by other researchers about the difficulty of conserving fitness score when the population has become fit. Based on the two observations, there is a strong indication that high success rates are connected with neutrality in its ability to defend against destructive mutations.

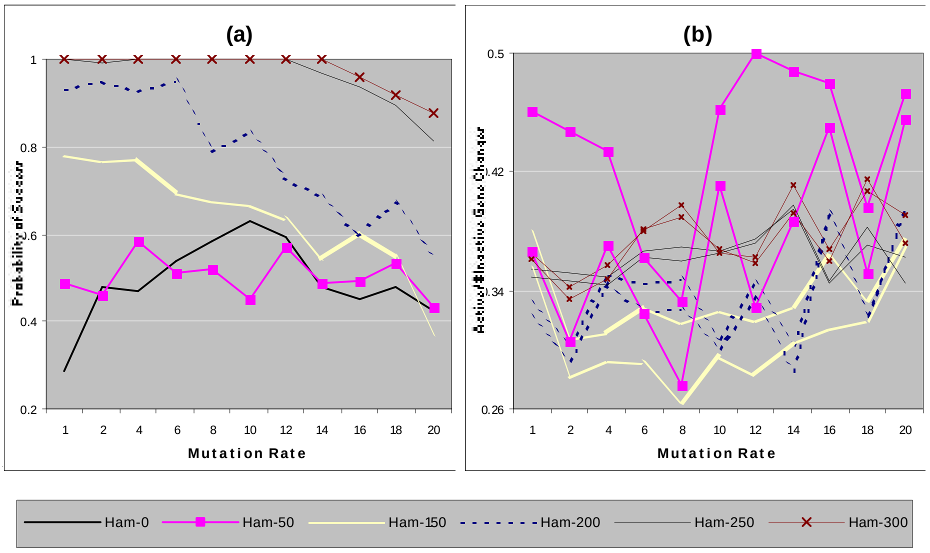


Figure 7 Fitness 7 and 8 Search Space.

Only hamming distance 250 and 300 implementations exhibit some of the exploitation/exploration ratios patterns observed in the other two search space. First, the two target ratios are close to each other. The ratio differences are around 0.006 and 0.008 respectively. However, these low ratio differences do not all correspond to 100% success rate. When mutation rate reaches 14%, the success rate starts to drop. We think this is due to the "rugged" landscape created by those mutation rate implementations, suggested by the uneven shape of the ratios curve with different mutation rates.

The hamming distance 200 implementation also has very small ratio difference (0.009). However, the successful rates are considerably worse than that of the hamming distance 250 and 300 implementations. We speculate that in addition to the "ruggedness" of the landscape, the range of the ratios also plays a role in the success rates. We have examined the magnitude of the exploitation/exploration ratios of those with 100% success rates and found that they are all between the range of 0.34 and 0.41. However, the hamming distance 200 implementation has higher inactive gene changes, i.e. more exploration. This might lead the evolution toward the areas where convergence is harder. The hamming distance 150 implementation also has a similar ratio pattern. Moreover, it has a larger target ratios difference. The experimental results show that it has an inferior success rate than that of the hamming distance 200 implementation.

Hamming distance 50 has a similar success rate to hamming distance 0. This indicates the neutrality (50) does not provide a sufficient buffer to protect against destructive mutations. The exploitation/exploration ratios suggest that the landscape is very rugged, which might be the reason of its inferior success rates.

In these experimental studies, we have identified the following features of exploitation/exploration ratios that are associated with high success rates:

- The ratios between neutral walk and fitness improvement are close to each other;
- The ratios are within a certain range of magnitude;
- The ratios increase as the mutation rates increase;
- The ratios increase in a very smooth manner.

7 Related Work

Sewall Wright addressed the question of to what extent can adaptive evolution benefit from neutral evolution in a letter to Kimura: "Changes in wholly non-functional parts of a molecule would be the most frequent ones but would be unimportant, unless they occasionally give a basis for later changes which improve function in the species in question which would become established by selection." [17] This view has been supported by recent studies on computer simulation of biological data and on the study of evolution of an artificial fitness landscape.

Using a standard folding procedure to map RNA sequences to its secondary structure (a many-to-one genotype-phenotype mapping process), [6] demonstrated that a neutral network can give rise to new previously unencountered phenotypes. Moreover, a selection induced transition can occur where two neutral networks are close at the genotypic level. Their study concluded that the success of the evolutionary search in such a space is less dependent on the initial conditions. In a different work, [14] shows directly that the maximum fitness obtainable during the adaptive walk of a population increases with increasing degree of neutrality.

Barnett extended the NK fitness landscape model to include neutrality where he observed that the "constant innovation" property helped to escape local optima [3]. Harvey and Thompson, in an experiment that used artificial evolution to configure a programmable silicon chip, also found that the presence of neutrality in a genetically converged population could lead to further fitness improvement [4].

The many-to-one genotype-phenotype mapping has also been studied as a mechanism of constraints handling in EC systems. In [2], Banzhaf focused on the mapping between unconstrained binary genotypes and constrained parse tree phenotypes In contrast, Yu and Bentley provided many different ways to evolve constrained phenotypes within the genotype-phenotype mapping framework [20]. Similarly, O'Neill and Ryan devised a Grammatical Evolution system to study genotype-phenotype mapping [15]. The investigation of neutral mutation on a symbolic regression and a Santa Fe ant trail problems is reported in [16].

8 Conclusions and Future Work

The quantitative study of the role of neutrality in EC systems is an important research subject. However, the lack of a measuring system has made the work difficult to conduct. In this paper, we have introduced the use of explicit neutrality with integer string coding to provide a mechanism for measuring neutrality during evolution. This approach has opened a door for a better understanding of the relationship between neutrality and the evolvability of EC systems.

Our initial study on the even-3 function landscape indicates that there is a positive relationship between neutrality and evolvability: *neutrality improves evolvability*, *neutrality improves success rates*. Moreover, the "buffer effect" of neutrality that protects genotypes from destructive mutations during later stage of evolution is strongly supported in our experimental results.

Our analysis, based on the probability of success and on the exploitation/exploration ratios, suggested an interesting interaction between neutrality and evolvability. When the adaptive and neutral mutations are maintained at a satisfactory level, evolution is able to generate fitter offspring. The characteristics of the satisfactory level of exploitation/exploration ratios are identified as:

- The ratios between neutral walk and fitness improvement are close to each other;
- The ratios are within a certain range of magnitude;
- The ratios increase as the mutation rates increase;
- The ratios are increased in a very smooth manner.

We will continue the investigation of these characteristics on other Boolean function problems to assess their validity and generality.

Acknowledgements

We would like to thank Vesselin Vassilev and Jennifer Powell for their suggestions. The resource support by CITC Math Modeling Disciplines is also greatly appreciated.

References

[1] Altenberg, L.: The evolution of evolvability in genetic programming. In: Advances in Genetic Programming, K. E. Kinner Jr., ed. MIT Press, (1994) 47-74.

[2] Banzhaf, W.: Genotype-phenotype-mapping and neutral variation: a case study in genetic programming. In: Proceedings of the Conference on Parallel Problem Solving from Nature III. Springer-Verlag, Berlin Heidelberg New York (1994) 322-332.

[3] Barnett, L.: Ruggedness and neutrality - the NKp family of fitness landscapes. In: Proceedings of the 6th International Conference on Artificial Life. MIT Press, (1998) 18-27.

[4] Harvey, I., Thompson, A.: Through the labyrinth evolution finds a way: a silicon ridge. In: Proceedings of First International Conference on Evolvable Systems: From Biology to Hardware, LNCS, Vol. 1259. Springer-Verlag (1996) 406-422.

[5] Hinton, G. E., Nowlan, S. J.: How learning can guide evolution. Complex Systems, Vol. 1, (1987) 495-502.

[6] Huynen, M. A., Stadler, P. F., Fontana, W.: Smoothness within ruggedness: the role of neutrality in adaptation. Proc. Natl. Acad. Sci. (USA) Vol. 93 (1996) 397-401.

[7] Kimura, M.: Evolutionary Rate at the Molecular Level. Nature, Vol. 217 (1968) 624-626.

[8] Kimura, M.: Some recent data supporting the neutral theory. In: New Aspects of the Genetics of Molecular Evolution. Springer-Verlag, Berlin (1991) 3-14.

[9] Kimura, M.: The neutral theory as a basis for understanding the mechanism of evolution and variation at the molecular level. In: Molecular Evolution, Protein Polymorphism and the Neutral Theory. Springer-Verlag, Berlin (1982) 3-56.

[10] Koza, J. R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press (1992).

[11] Miller, J. F.: An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming Approach. In: Proceedings of the First Genetic and Evolutionary Computation Conference (GECCO'99), Morgan Kaufmann, San Francisco, CA (1999) 1135-1142.

[12] Miller, J. F., Thomson, P., Fogarty, T. C.: Designing electronic circuits using evolutionary algorithms, arithmetic circuits: a case study. In: Genetic Algorithms and Evolution Strategies in Engineering and Computer Science, Wiley, Chichester, UK (1998) 105-131.

[13] Miller, J. F., Thomson, P.: Cartesian genetic programming. In: Proceedings of the Third European Conference on Genetic Programming (EuroGP2000). Lecture Notes in Computer Science, Vol. 1802, Springer-Verlag, Berlin (2000) 121-132.

[14] Newman, M. E. J., and R. Engelhardt, R., Effects of neutral selection on the evolution of molecular species", Proc. Roy. Soc. London Series B Vol. 265 (1998) 1333-1338.

[15] O'Neill, M., Ryan, C.: Under the hood of grammatical evolution. In: Proceedings of the First Genetic and Evolutionary Computation Conference (GECCO'99), Morgan Kaufmann, San Francisco, CA (1999) 1143-1148.

[16] O'Neill, M., Ryan, C.: Genetic code degeneracy: implications for grammatical evolution and beyond. In: Proceedings of the 5th European Conference on Artificial Life, (1999).

[17] Provine, W. B.: Sewall Wright and Evolutionary Biology. The University of Chicago Press, Chicago (1986).

[18] Schwefel, H.P.: Kybernetische Evolution als Strategie der experimentellen Forschung in der Stromungstechnik. Diplomarbeit, Technische Universitat Berlin, 1965.

[19] Vassilev, V. K., Miller J. F.: The advantages of landscape neutrality in digital circuit evolution. In: Proceedings of the 3rd International Conference on Evolvable Systems: From Biology to Hardware, Lecture Notes in Computer Science, Vol. 1801, Springer-Verlag, Berlin (2000) 252-263.

[20] Yu, T. and Bentley, P.: Methods to evolve legal phenotypes. In: Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature. Lecture Notes in Computer Science, Vol. 1498, Springer-Verlag, Berlin (1998) 280-291.