Through the Interaction of Neutral and Adaptive Mutations, Evolutionary Search Finds a Way

Tina Yu
Department of Computer Science
Memorial University of Newfoundland
St. John's, NL A1B 3X5
Canada
tinayu@cs.mun.ca

Julian Francis Miller
Department of Electronics
University of York
York YO10 5DD, UK
jfm@ohm.york.ac.uk

Abstract An evolutionary system that supports the interaction of neutral and adaptive mutations is investigated. Experimental results on a Boolean function and needle-in-haystack problems show that this system enables evolutionary search to find better solutions faster. Through a novel analysis based on the ratio of neutral to adaptive mutations, we identify this interaction as an engine that automatically adjusts the relative amounts of exploration and exploitation to achieve effective search (i.e., it is self-adaptive). Moreover, a hypothesis to describe the search process in this system is proposed and investigated. Our findings lead us to counter the arguments of those who dismiss the usefulness of neutrality. We argue that the benefits of neutrality are intimately related to its implementation, so that one must be cautious about making general claims about its merits or demerits.

Keywords
Neutrality, exploration, exploitation, adaptive mutations, neutral mutations, genetic programming, self-adaptation

1 Introduction

The field of evolutionary computation (EC) has drawn much of its inspiration from natural evolution. The concepts of population, selection, and mutation have been implemented (with variation) in many EC systems to carry out computing tasks, such as optimization and modeling. One area that has not received much attention is the utilization of random genetic drift (through neutrality) based on a biologically motivated encoding. DNA in organisms consists of protein-coding regions and also noncoding regions (commonly referred to as junk DNA). Protein-coding regions consist of coding regions (instructions for making proteins) and promoter and enhancer regions. The promoter region is essentially a start instruction for a molecule to begin a transcription of the DNA sequence into RNA, an intermediate language that will be translated into protein later. The transcription of genes in DNA into RNA can be either stimulated or inhibited by other molecules that bind to promoter or enhancer regions [5]. In other words, some of the DNA code can be switched on while some other parts are switched off.

This article investigates random genetic drift (neutrality) that occurs in an EC system called Cartesian Genetic Programming (CGP). Like the biological genome, a genotype in CGP has coding (active) and noncoding (inactive) regions. Moreover, the coding region contains instructions and promoters (links). The translation of instructions to its phenotype is under the control of promoter genes. This genotype representation leads to the interaction of neutral and adaptive mutations during the evolutionary process (see Section 4.4), in accord with a view of natural evolution expressed by Sewall Wright (see Section 2). We have tested this system on two types of search problems. The first is a Boolean function problem, and the second, a needle-in-haystack fitness landscape. Our experimental results show that this genotype representation, through the action of genetic drift, generates evolutionary dynamics that enables better solutions to be found faster. Although the two problems seem small, they represent two very important types of fitness landscapes. The reported results therefore are significant.

Other work has been published that investigates the behavior and performance of CGP on larger-scale problems. For example, Miller and colleagues studied digital circuit design [22], and Garmendia-Doval and Miller worked on molecular screening [23]. Both works showed that large neutral genetic drift was extremely beneficial; this accords with our findings in this study.

The organization of the article is as follows. Section 2 gives the biological background that motivates this research. Section 3 discusses the concept of neutral evolution in EC. In Section 4, the evolutionary system is described. Following that, we review related published work in Section 5. Section 6 presents the experimental results of a Boolean function problem, and Section 7 gives our study of a set of needle-in-haystack problems. In Section 8, we discuss some open issues on neutrality in EC. Finally, Section 9 concludes the article and provides avenues for future research.

2 Background and Motivation

The theory of evolution by natural selection 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 this purely selectionist view was challenged. In particular, Motoo Kimura observed that the number of mutant substitutions in amino acid sequences of hemoglobin was too large to be explained by the theory of evolution by natural selection. Based on this discrepancy, he proposed the *neutral theory*, which is founded on the premise that most mutations at the molecular level in evolution are caused by random genetic drift rather than purely by natural selection [13, 14]. In other words, the mutations involved are neither advantageous nor disadvantageous to the survival or reproduction of the individual. This idea has provoked much controversy (often called neutralist-selectionist controversy) and is still a subject of strong debate [17].

In 1973, Tomoko Ohta, an associate of Kimura, started articulating an alternative to the strictly neutral and strictly selectionist models. In her *nearly neutral theory*, she describes a class of mutations ("borderline mutations") that are influenced by both genetic drift and selection [25, 26]. Moreover, the theory states that in small populations, mutations are characterized by random drift. In large populations, they are governed by selection. To test this theory, she studied the ratio of adaptive to neutral mutations in DNA sequence data. In particular, she compared synonymous and nonsynonymous mutations of 49 mammalian genes. A mutation is said to be synonymous if it causes no change in the amino acid specified (e.g., codons GTC and GTA both code for the amino acid valine); otherwise it is called nonsynonymous. Her results support the theory [27]. However, a different study measuring the relative numbers of synonymous and nonsynonymous mutations within a species gave results that contradict the theory [28]. Hence, the issue remains unsettled.

Sewall Wright was one of the first to observe that genetic drift could play an important role in the process of evolution [30]. When asked about the extent to which adaptive evolution can benefit from neutral evolution, he replied to Kimura in a letter: "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" [30]. This view has been supported by recent studies on computer simulation of biological data [12] and on the study of evolution on an artificial fitness landscape [24]. In this article we are interested in an implementation of genetic drift that accords with Wright's view of evolution.

3 Neutral Evolution in Evolutionary Computation

EC systems model the process of natural evolution for problem solving by generating a population of possible solutions, which are selected and mutated to reach a near-optimum. In an EC system, a genotype represents a point in the search space and is operated on by genetic operators. In contrast, a phenotype represents a point in the solution space and is evaluated by the fitness function. In other words, selection is based on phenotypes, while reproduction operates on the underlying genotypes [39]. Although some EC systems—for example, standard genetic programming (GP) [16]—do not distinguish genotypes from phenotypes, more and more EC systems are designed with this dual representation and to that extent are closer to biological systems.

One advantage of the dual representation is that it provides a mapping step. When there exists a many-to-one mapping between genotypes and phenotypes, a mutation operation may transform genotypes from one to another without affecting the phenotype (thus having the same fitness). In other words, the mutation is neutral. With the dual representation, we can then distinguish two sources of neutrality: many genotypes may map to the same phenotype, and many phenotypes may evaluate to the same fitness.

3.1 Implicit Neutrality

Neutrality embedded in the genotype representation is implicit. For example, in the standard GP representation there are two forms of implicit neutrality: functional redundancy and introns. Functional redundancy refers to the case when many different programs (genotypes) represent exactly the same function (phenotype). For instance, the following three programs give the same function XOR:

G1: NOR $(\text{AND } x_1 \; x_2)(\text{NOR } x_1 \; x_2)$

G2: NOR $(\text{NOR } x_1 \; x_2)(\text{AND } x_1 \; x_2)$

G3: NOR $(\text{AND } x_1 \; x_2)(\text{NOR } (\text{NOR } x_1 \; x_2) \; (\text{NAND } x_1 \; x_2))$

Genetic transformation from one genotype to another (e.g., G1 to G2) has a neutral effect on the program's behavior. Functional redundancy is semantic. Hence the locations of neutral mutations are not easy to identify.

Introns are code that is part of a program but is semantically redundant (i.e., do not affect program behavior). For example, the function OR with input FALSE is an intron in the following genotype because (OR FALSE ANYBOOLEAN-VALUE)=ANYBOOLEANVALUE:

G4: OR FALSE $(\text{NOR } (\text{AND } x_1 \; x_2)(\text{NOR } x_1 \; x_2))$

Genetic transformations that remove introns from a genotype (e.g., G4 to G1) have a neutral effect on the program's behavior. As with functional redundancy, introns in GP are also semantic. They are redundant because of the context they are in. When the code is moved to a different place, it may no longer serve as an intron.

Functional redundancy and introns can emerge within an evolving genetic program. As a consequence, they are not easy to identify or control during the program evolution.

3.2 Explicit Neutrality

Explicit neutrality is associated with extra code in the genotype that is not expressed in the phenotype. Unlike implicit neutrality, which is semantic, explicit neutrality is syntactic. Physically, there is extra code in the genotype, in which neutral mutations may reside.

A genotype representation that includes explicit neutrality has two parts: active and inactive; only genes that are active are expressed in the phenotype. For example, the following two genotypes have extra code that is inactive (underlined). Since their active parts are identical, they are mapped into the same phenotype (XOR function):

G5: NOR (AND x_1 x_2) $\underline{\text{(NOR } x_1\ x_2\text{)}}$ (NOR x_1 x_2)

G6: NOR (AND x_1 x_2) $\underline{\text{(AND } x_1\ x_2\text{)}}$ (NOR x_1 x_2)

Genetic changes on inactive genes (e.g., G5 to G6) have a neutral effect on the program's behavior. Genotypes with inactive genes, therefore, can provide additional neutrality during the evolutionary process.

In this work, we investigate the properties of a genotype representation that allows both implicit and explicit neutrality. This program representation and the neutrality it allows are described in the following section.

4 Wright-Inspired Neutrality Model

In a fashion somewhat similar to the representation of biological genomes, CGP [20] allows neutral code that may be selected for later by adaptive mutations.

4.1 Genotype and Phenotype Representation

The genotype represents a graph that was originally developed for evolving digital circuits [21]. A similar representation of programs (called Parallel Distributed GP) was independently devised by Poli [29]. In essence, the encoding is a string of integers that represent the functions and links (connections) between graph nodes and graph inputs and outputs. This gives it great generality, so that it can represent neural networks, programs, circuits, and many other computational structures. Although, in general, it is capable of representing directed multigraphs, it has so far only been used to represent directed acyclic graphs, including connected and non-connected graphs. In a non-connected graph, one or more nodes cannot be visited by following the directed links. Such unvisited nodes are genetic material that is not utilized in the phenotype, and is analogous to noncoding DNA.

In its original formulation, the genotype was represented as a directed Cartesian grid of nodes in which nodes were arranged in layers (rows) and it was necessary to specify the number of nodes in each row and the number of columns in the grid (see Figure 1). The nodes in each column were not allowed to be connected to nodes in different rows (rather like a multilayer perceptron neural network). In addition, a parameter was introduced called *levels-back,* which specified how many columns back a node in a particular column could connect to. The graph inputs were specified in the *input layer* on the left of the array of nodes.

It is important to note that in many implementations of CGP (including this one), the number of rows (r) is set to one. In this case the number of columns (c) becomes the *maximum* allowed number of nodes (user-defined). Also the parameter *levels-back* can be chosen to be any integer from one (in which case, nodes can only connect to the previous layer) to the maximum number of nodes (in which case a node can connect to any previous node). It should be noted that the output genes can be dispensed with (as in this study) by choosing the program outputs to be taken from the m rightmost consecutive nodes (when only one row is used). Here we only consider programs with a single output.

The Cartesian genotype is a string of integers as follows:

$$C_0, f_0, C_1, f_1, \ldots, C_{cr-1}, f_{cr-1} O_0, O_1, \ldots, O_m$$

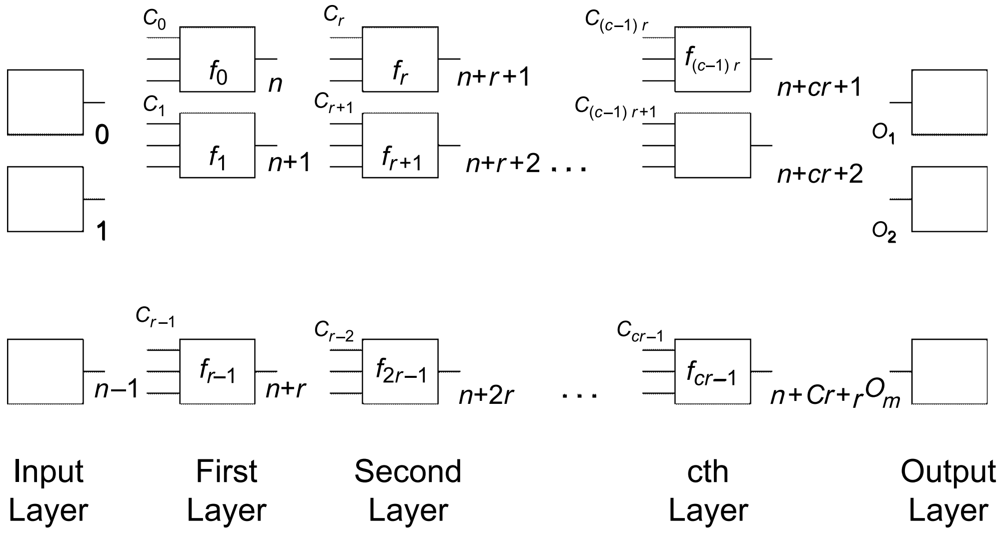

Figure 1. General form of Cartesian genotype for an n-input m-output function. There are three user-defined parameters: number of rows (r), number of columns (c), and *levels-back* (see text). Each node has a set of C_i connection genes (according to the arity of the function) and a function gene f_i that defines the node's function from a lookup table of available functions. On the far left are seen the genotype inputs, or terminals, and on the far right the genotype output connections O_i.

Here, C_i denotes a set of connection points of a node that provide inputs to the node function f_i, and O_i are the output values of the graph. The available node functions are defined by the user. Sometimes it happens that not all node functions have the same arities (so the cardinality of C_i varies). Usually this is handled by setting the node arity to the maximum arity that appears in the function set. Nodes with functions that require less inputs than the maximum ignore the extra inputs.

In a directed acyclic graph, the range of allowed alleles for the node connection genes C_i is restricted so that nodes can only have their inputs connected to either an input node or a node from one of the previous (left) columns. Also, function values are chosen from the set of available functions.

For example, Figure 2 shows a genotype and its phenotype for an even-5-parity function (described in Section 6). The genotype has six nodes; each has three genes (two input link values and one Boolean function value). The Boolean function (in **bold**) takes values from the set {0,1}, which represent the Boolean function XOR and EQ, respectively. The input link values can be either an input to the even-5-parity function, denoted by labels 0 to 4 in the genotype (and as x_0 to x_4 in the phenotype), or a node output link, denoted by a label from 5 to 9. The last node (with label 10) is the final output node.

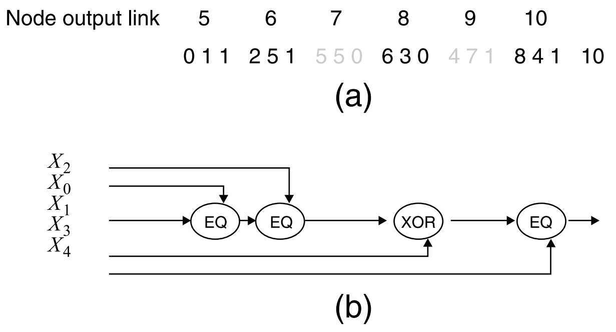

Figure 2. A genotype (a) and its phenotype (b) for even-5-parity.

The mapping of a genotype to its phenotype starts from the last node on the right. This is the final output node (with genes 8 4 1), which is active by default. This EQ node has one input connected the even-5-parity input number 4, and the other connected to the node with output link 8. This makes node 8 active. The node with output link 9 is not expressed in the phenotype, because it is not referenced by any active nodes. The same applies to the node with output link 7. These two inactive nodes are in gray in Figure 2. All other nodes are referenced by one of the active nodes, so they are expressed in the phenotype.

This genotype-phenotype representation creates some unique evolutionary dynamics: active genes may become inactive and vice versa. This happens when mutations are applied to link values. For example, if one link value in node 10 is mutated from 8 to 7, it causes two active nodes (8 and 6) to become inactive, and one inactive node (7) to become active (see Figure 3). In other words, neutral mutations (inactive genes) may later be selected (when they become active) for adaptive mutations. The dynamics of this is much like the interaction of neutral and adaptive mutations described in Wright's letter. Note that the mutation operator performs type checking: links can only be changed to valid links, and functions to valid functions.

Another important characteristic of this representation is that it provides a systematic way to measure neutrality. We described this in the following subsection.

4.2 Measuring Neutrality

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 are associated with implicit neutrality. In contrast, neutral mutations on inactive genes are associated with explicit neutrality. Regardless of the source of neutrality (implicit or explicit), the overall amount of neutral mutations between the parent-offspring genotype pair (*g, h*) can be measured according to their Hamming distance *H*(*g, h*):

$$H(g,h) = \sum_{i=1}^{\mathrm{len}(g)} \delta_{g_i h_i}$$

where len(*g*) is the number of genes in the genotype *g*. The Kronecker delta δ is defined as follows:

$$\delta_{mn} = \begin{cases} 1, & m \neq n \\ 0 & \text{otherwise} \end{cases}$$

Let us define the activity *a*(*g*$_i$) of a gene *i* in genotype *g* as follows:

$$a(g_i) = \begin{cases} 1, & g_i \text{ active} \\ 0 & \text{otherwise} \end{cases}$$

Node output link	5	6	7	8	9	10	
	0 1 **1**	2 5 **1**	5 5 0	6 3 **0**	4 7 1	8 4 **1**	10
	0 1 **1**	2 5 1	5 5 **0**	6 3 0	4 7 1	7 4 **1**	10

Figure 3. The interaction of adaptive and neutral mutations.

We can then define the number of active gene changes, $H_a(g,h)$, as follows:

$$H_a(g,h) = \sum_{i=1}^{\mathrm{len}(g)} \delta_{g_i h_i} a(g_i) a(h_i) + \delta_{a(g_i)\overline{a(h_i)}} + \delta_{\overline{a(g_i)}a(h_i)}$$

The first term in the equation is the number of genes that are active in both genotypes g and h [$a(g_i) = a(h_i) = 1$] but with different values ($\delta_{g_i h_i} = 1$). The second term is the number of genes that are active in genotype g but inactive in genotype h, and the third item is the number of genes that are inactive in genotype g but active in genotype h. Here, the overbars on a_i mean logical inversion.

Similarly, the number of inactive gene changes, $H_i(g, h)$, is defined by

$$H_i(g,h) = \sum_{i=1}^{\mathrm{len}(g)} \delta_{g_i h_i} \overline{a(g_i)}\ \overline{a(h_i)}$$

For example, the two genotypes in Figure 4 have a Hamming distance of 6. The number of active gene changes is 12 (each of the nodes numbered 6, 7, and 8 contributes three active gene changes, because they were switched either from active to inactive or from inactive to active). The number of inactive gene changes is one; this takes place on node number 9.

4.3 Evolutionary Algorithm

We have adopted a simplified evolution strategy [32] for evolutionary search; one parent with four offspring (1+4; population size 5). This algorithm was shown to be computationally more efficient [22]. It has been used widely by the second author and coauthors. Part of the intention of this study is to shed light on how such a simple evolutionary algorithm can be so effective for search. The algorithm is described as follows:

1. Generate an initial population of five individuals randomly.
2. Evaluate the fitness for each individual in the population.
3. Select the best of the five in the population as the winner.
4. Carry out pointwise mutation on the winning parent to generate four offspring.
5. Construct a new generation with the winner and its four offspring.
6. Select a winner from the current population, using the following rules:
   (a) If there are offspring that have better fitness than the parent has, the best offspring becomes the winner.
   (b) Otherwise, if there are offspring that have the same fitness as the parent, then one is selected randomly. If the parent-offspring genotype pair has a Hamming distance that is within the permitted range, the offspring becomes the winner.
   (c) Else the parent remains as the winner.

Node output link	5	6	7	8	9	10	
G7:	0 1 **1**	2 5 **1**	5 5 0	6 3 **0**	4 7 1	8 4 **1**	10
G8:	0 4 **1**	2 5 1	5 5 **1**	5 3 0	4 2 1	7 3 **1**	10

Figure 4. Measuring the Hamming distance between two genotypes.

7. Go to step 4 unless the maximum number of generations has been reached.

In step 6(b), the Hamming distance between a parent-offspring pair is used to specify the degree of neutrality allowed in the evolutionary search. If the permitted distance is 0, neutrality is not allowed; only better offspring are accepted to replace the parent as the winner. In other words, evolution has to constantly provide fitness improvement in order to proceed. To grant neutrality, a non-0 Hamming distance is used. The larger the permitted Hamming distance, the larger the amount of neutral mutation that is allowed during evolutionary search. When the permitted Hamming distance is the same as the length of the genotype, evolution can freely undergo unrestricted genetic drift until an improved offspring is encountered.

4.4 Exploitation versus Exploration

Mutations on a genotype that has some of its genes active and others inactive produces various effects. Mutations on active genes can be *exploited* if they confer fitness improvement. Mutations on genes that were inactive and become active or were active and become inactive are *exploitable,* since they switch gene expression and alter the phenotype. In contrast, mutations on inactive genes that remain inactive *explore* new genetic material and serve to preserve genetic diversity. Based on this view, evolutionary search is a process of exploiting known information through mutations on active genes and exploring new information through mutations on inactive genes. We capture the two aspects of evolutionary search with a ratio of exploitation to exploration by calculating the numbers of active and inactive gene changes:

$$\frac{\text{no. of active gene changes}}{\text{no. of inactive gene changes}} = \frac{\text{exploitation}}{\text{exploration}}$$

In this study we have calculated this ratio of exploitation to exploration during evolutionary search and have investigated how it changes under a variety of conditions. We will show later that there is a strong correlation between the value of this ratio and the success of evolutionary search. Moreover, the ratio can be automatically maintained (self-adaptive) throughout the evolutionary process. While the design of an effective evolutionary algorithm that provides a balanced exploitation and exploration is still a challenge [2, 3], our work here indicates that the simple evolutionary strategy using the CGP genetic representation is a particularly effective search technique. This argument will be justified through a series of exhaustive experiments and analysis in Sections 6 and 7.

4.5 Neutral Walk versus Fitness Improvement

With the presence of neutrality and an elitist selection scheme, the evolutionary search is a sequence of neutral walks and fitness improvements. As described in step 6 of Section 4.3, the best-so-far individual is kept as the current winner. Moreover, the current winner can only be replaced by another genotype with *equal* or *better* fitness. When the winner is replaced by a genotype with the same fitness, evolutionary search is undergoing a 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 a neutral walk and the search for a fitter genotype.

During the neutral walk, mutations on active genes exploit implicit neutrality, while mutations on inactive genes explore explicit neutrality. For example, in Figure 5, both genotypes G1 and G2 represent the Boolean function XOR. A two-point mutation on G1 produces G2 without changing the semantics (fitness) of the phenotype. This mutation, however, causes eight active gene changes, and thus exploits implicit neutrality. Although they did not give fitness improvement, previously unexpressed genes have become active.

The exploitation of implicit neutrality and exploration of explicit neutrality during neutral walk may or may not lead to a fitter genotype, depending on how much exploitation and exploration is

Node output link	3	4	5	6	7	
G1:	1 2 3	1 2 1	1 2 4	3 4 3	4 5 4	7
G2:	1 2 3	1 2 2	1 2 4	3 4 3	6 5 4	7

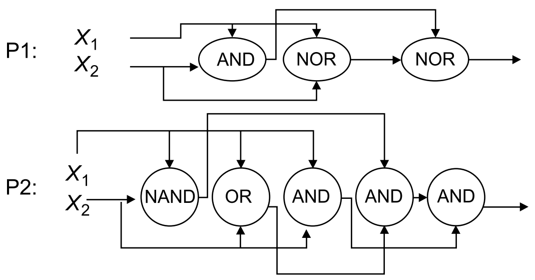

Figure 5. Mutations on active and inactive genes during neutral walk.

carried out. The relationship between the exploitation/exploration ratio and the probability of fitness improvement is the core of this work. We have conducted two sets of experiments: one on a Boolean function landscape, and one on a needle-in-haystack landscape. Sections 6 and 7 will give detailed analysis. Based on our findings, we will propose a hypotheses that describes the evolutionary search process under this neutrality model (Section 8).

5 Other Studies of Neutrality in EC

Other neutrality models have been studied in EC. We review these works in the following subsections.

5.1 Potentially Useful Junk: Redundancy in Hardware Evolution

Harvey and Thompson argued that not all neutrality is helpful [10]. For example, simply adding extra redundant loci in the genotype would make multiple genotypes represent the same phenotype. However, this type of redundancy is *useless junk*. Nothing will be gained by this exercise. Forrest and Mitchell [8] have implemented a similar kind of junk (noncoding segments) in their GA. They reported that noncoding segments do not improve their GA performance. This result is confirmed by Wu and Lindsay [34] (see Section 5.4).

Potentially useful junk, defined by Harvey and Thompson [10], consists of "loci on a genotype which within the current context of the rest of the genotype are functionless, but given different values elsewhere on the genotype they may well become significant." They implemented this kind of junk by providing more components than needed in the genotype for hardware (FPGA) evolution. The genotype of their Species Adaptation Genetic Algorithm (SAGA) [11] system has a two-dimensional layout. Each component may be wired to its four neighboring components. Those that are not wired to the functional part but are peripheral to the functional part are potentially useful junk, as they may be wired to the functional parts as evolutionary search progresses (e.g., by mutation). Those components that are too far away from the function part cannot be linked together, and hence are not useful.

CGP also provides the described potentially useful junk: inactive (nonfunctional) genes may become active (functional). In the CGP system studied here, each node in the graph may be connected to any nodes on its left. In contrast, a cell in a SAGA genotype is constrained to connect only to its four neighboring cells. In other words, a CGP genotype may have a higher-dimensional topology than a SAGA one. On the one hand, CGP gives a higher probability to switch nonfunctional genes into functional—that is, all redundant genes are potentially useful junk. On the other hand, it also introduces more linkages between genes. How the dynamics of these two effects influence the evolutionary search is unknown and requires further study.

Harvey and Thompson also investigated how the search performance is influenced by the presence of neutrality. They believed that the effectiveness of the search depends on balanced selection and mutation: an optimum occurs at the point where mutation gives sufficient diversity but not so much as to cause a loss of useful accumulated genetic information (they called this an *error catastrophe*). With a population size of 50 and a rank-based selection scheme, they estimated that mutating 2.7 bits in a genotype of 1800 bits is safe (i.e., less than the error catastrophe) provided that at least 63% of the genotype was redundant. No performance comparison on genotypes with neutrality and without neutrality was conducted.

We also studied the interplay between neutrality and mutation rate in our work. Instead of checking the existence of an error catastrophe, our analysis is based on the ratio of active to inactive gene changes. This study will be discussed in Section 6.

5.2 Many-to-One Genotype-Phenotype Mapping

Ebner and colleagues investigated neutrality through three genotype-phenotype mapping schemes that provide different amounts of genotype redundancy [6]. A binary mapping is a one-to-one mapping that does not have genotype redundancy. A cellular automaton mapping gives a genotype redundancy ratio of 2^{64} to 1. A random Boolean network mapping gives the highest genotype redundancy: each phenotype is represented by 2^{136} genotypes.

They compared these three mapping methods on three different criteria: (1) the number of other phenotypes that a phenotype can reach using one-point mutation within 100 time steps, (2) the innovation rates, and (3) the connectivity between phenotypes. They reported that the mappings with a higher degree of redundancy allow phenotypes to be more reachable via single-point mutation. They also applied these mapping methods to two problems: one with a static fitness landscape and the other with a dynamic landscape. For the first problem, they used a population size of 100 with a single-point mutation operator. An offspring with equal or better fitness replaces the parent. Essentially they are using a population-based hill-climbing algorithm with neutrality. Since there is no interaction between individuals (no crossover or competition), each individual in the population is carrying out the hill climbing independently. A experimental run is equivalent to 100 independent runs of a 1+1 hill-climber with neutrality. For the second problem, they added crossover. They reported that in both experiments, higher average population fitness is obtained by the mapping with the highest degree of genotype redundancy [7].

Recent work by Knowles and Watson argues that the claimed exploration through many-to-one genotype-phenotype mapping can also be achieved through higher mutation rates [15]. They used a GA with steady-state elitism and higher mutation rates ($1/l$, $2/l$, $4/l$, and $8/l$, where l is the length of the genotype) to conduct their experiments. They also use the same mapping methods as were used in [6]. With the NK2 problem, they reported that both many-to-one and one-to-one mappings were able to find the solutions, yet a one-to-one mapping does that with a lesser number of iterations (note that the steady state does not have the concept of generation). They also conducted experiments on other problems (NK4, SAT32, SAT64, and H-IFF). They compared the fitness of *final solutions* and reported that no significant improvement in final fitness was observed using many-to-one mapping.

In a recent investigation of redundant representations, Rothlauf and Goldberg [31] distinguish two classes of redundant representations. The first one was called synonymous redundancy. In this class *all* genotypes that are mapped to the same phenotype are genotypically close (i.e., their Hamming distance is small). The trivial voting mapping is an example of a synonymously redundant mapping (a majority vote is carried out over a neighborhood of binary genotype bits to determine the phenotype bit). If this is not the case, then the representation belongs to the second class, which they refer to as nonsynonymous. They argued that nonsynonymous redundant mappings had little utility in search problems; however, they gave no mathematical analysis to support this assertion. Instead they only analyzed synonymously redundant representations. Some of the mappings investigated by Ebner et al. [6] were also nonsynonymous, with different amounts of redundancy (one based on cellular automata, the other on random Boolean networks).

It is easy to see that the CGP genotype representation is nonsynonymous according to Rothlauf and Goldberg's definition. Figure 6 gives two genotypes, G11 and G12, where only the rightmost node with genes 010 is active. Hence, they have the same phenotype. However, the two genotypes are very different (in fact, 15 of the 18 genes are different). The genes that are different in the two genotypes are inactive (gray), so whatever alleles they have make no difference to the phenotype. We will demonstrate through experimental results that despite the nonsynonymous nature of the representation, it is highly beneficial to the evolutionary search. This disagrees with Rothlauf and Goldberg's assertion about the limited utility of nonsynonymous mappings and shows that the issue is more complex and requires more research.

5.3 Many-to-One Genotype-Fitness Mapping

Another way to induce a neutral network in a genotype space is to *coarse-grain* the fitness value into a small number of fitness classes. Under such coarse graining, genotypes whose fitness fall into the same fitness class are treated as mutually neutral under selection. For example, neutral networks in *NK fitness landscapes* have been constructed this way by Newman and Engelhardt [24]. In contrast, van Nimwegen and Crutchfield [35] defined a class of *royal staircase* fitness functions that also lead to many-to-one genotype-fitness mapping. They demonstrated that the performance of an evolutionary algorithm for this class of fitness function depends on factors such as mutation rates, population size, and the parameters specifying the fitness function.

5.4 Noncoding Segments in Genetic Algorithms

Levenick introduced the idea of inserting noncoding segments in a GA representation [19]. Unlike inactive genes in CGP, which may become active (and vice versa) as the result of evolutionary dynamics, noncoding genes remain silent throughout the evolutionary process. He tested this coding scheme on a simple five-exon problem (each exon having six genes) and reported that they improved search success rates. Although he proposed a couple of explanations of why they did so, 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 taken to find the optimum) by including noncoding segments in their GA [8].

Wu and Lindsay repeated the same experiments of Levenick and of Forrest and Mitchell. They reported similar results to those reported by Forrest and Mitchell [34]. Additionally, they extended their experiments on the royal road function 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. They reported that noncoding segments impair the GA performance (in terms of the number of generations taken to find the optimum) in the first case, but improve the performance in the second case.

The linkage-learning GA [9] also has a representation that contains links that to some extent are similar to those in CGP. In this GA, each gene has two kinds of values: function and location (link). However, unlike those in CGP, genes with duplicated locations are deleted from the genotypes. Such duplications are normally the result of crossover and mutation. Instead of retaining them as inactive genes (which CGP does), these redundant genes are removed. As a result, no explicit neutrality is supported in the linkage-learning GA.

Node output link	2	3	4	5	6	7
G1	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 1 0
G2	1 1 1	1 1 1	1 1 1	1 1 1	1 1 1	0 1 0

Figure 6. An example showing that a genotype is nonsynonymous.

5.5 Avida Artificial Life System

Claus Wilke and colleagues studied the evolution of digital organisms (as computer programs) using the Avida system [33]. They reported that under high mutation rates, an organism that has neighbors (those accessible by one mutation step) with a similar fitness (not necessarily the same fitness) has a higher reproduction rate. The reason is that such a flat fitness landscape is more robust against mutations than a fitness landscape that has high and narrow peaks. Although they did not mention neutral networks (where the neighbors have the same fitness), one would expect similar findings for them.

Their result is perhaps not surprising, as there have been various research reports on the buffering effect of neutral mutations against disruptive mutations [1, 38, 36]. The existence of neutral networks hence is beneficial to the organism's reproduction rate. We have reported a similar result on a Boolean function landscape in our previous work [40]. However, with needle-in-haystack landscapes, the opposite result is found: high mutation rates also give high reproduction rates [41]. The details will be discussed in Sections 6 and 7.

6 Experiments on a Boolean Function Landscape

Even-3-parity is a Boolean function that takes three Boolean inputs and returns TRUE only if an even number of inputs are TRUE. Table 1 summarizes the parameters used to conduct the experiments. 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 input combinations, hence eight test cases. The fitness of a program is the number of test cases in which it gives the correct output; thus, a program can have fitness between 0 and 8. The genotype has 100 nodes; each has three genes (two input link values and one Boolean function value). The length of the genotype is therefore 300. The mutation rate is the percentage of the genes in the genotype that are modified in one mutation operation. For example, mutation at rate 1% changes three genes in one operation (note that after a mutation site is selected, it may be mutated into itself). The neutrality level is the Hamming bound used in the selection step of the evolutionary algorithm (see Section 4.3). To obtain statistically meaningful results, we made 100 runs for each mutation rate with each neutrality level. Moreover, each run continued until the maximum number of generation was reached.

Table 1. Experimental setup for even-3-parity.

Description	Parameters
Function set	AND, NAND, OR, NOR
Terminal set	x_0, x_1, x_2
No. of test cases	8
Possible fitness	0, 1, 2, 3, 4, 5, 6, 7, 8
Genotype length	300 genes (100 nodes)
Mutation rates (%)	1, 2, 6, 10, 12, 14, 16, 18, 20
Neutrality level	Ham-0, Ham-50, Ham-150, Ham-200, Ham-250, Ham-300
Max. no. of generations	10,000
No. of runs	100

6.1 Results

The search performance is assessed on the population evolvability, defined as the ability of a population to produce variants fitter than any yet existing. We used the *best fitness of the population* as the indication of evolvability. Figure 7 plots the best fitness during the runs under eight different mutation rates and six different neutrality levels. At a 1% mutation rate, there is a clear separation of performance for each Hamming level. Hamming bound zero (no neutral drift) has the worst search performance. Once the Hamming bound is relaxed, the fitness improves rapidly. The performance reaches its best at Hamming bounds 250 and 300.

As the mutation rate increases, the performance gap between different Hamming bounds becomes smaller. Nevertheless, the search performance with Hamming bounds 250 and 300 remains the best of all. Meanwhile, the search performance with Hamming bounds 200, 150, and 50 indicates that the higher the Hamming bound, the better its final fitness. In other words, by supporting more freedom for genetic drift, evolution can generate fitter offspring; that is, the type of neutrality incurred by this genotype representation improves evolvability. This pattern, however, is not observed in Hamming bound 0 results, which are highly sensitive to the mutation rates. With mutation rates of 2% and 6%, Hamming bound 0 has similar results to Hamming bounds 50 and 150. With a mutation rate of 8%, Hamming bound 0 has a result that is better than that of Hamming bound 50 but worse than that of Hamming bound 150. When the mutation rate is increased to 10%, Hamming bound 0 outperforms both Hamming bounds 50 and 150. When the mutation rate is higher than 10%, the performance of Hamming bounds 0, 50, 150, and 200 becomes similar (200 being always superior). This suggests that under high mutation rates, high Hamming bounds (large neutral drift) are necessary to improve the evolutionary search. It will be shown in Section 6.3 that this is due to mutations becoming destructive when the population is fit. To continue improving the fitness, evolutionary search requires high neutrality to defend against a large number of mutations.

It is interesting to contrast these results with those of Knowles and Watson [15]. They found that the search performance (in terms of the final best fitness) of many-to-one genotype-phenotype mappings was not superior to that of a one-to-one (direct) mapping when one could choose from the best of a range of higher mutation rates through a trial-and-error series. They saw many-to-one mapping as providing a greater exploratory power than direct mapping at low mutation rates. At higher mutation rates, however, a direct mapping was superior in that it could find optima faster than a many-to-one mapping. Our results show very different behavior. We find that the best performance is achieved with the *maximum* allowed neutral walk under *all* mutation rates studied (going up to 20%). Moreover, when insufficient neutrality (Hamming bound 50, 150, or 200) is used under high mutation rates, no significant effect on evolutionary search is observed.

In contrast, the active and inactive gene interaction in CGP always allows unrestricted genetic drift when a Hamming bound is used that is equal to the length of the genotype. Although it may be more than what is needed (e.g., Hamming bound 250 is sufficient for this Boolean function problem), having a maximum Hamming bound does no harm to the performance, a result that agrees with the findings of [37]. This example demonstrates that different types of genetic redundancy can have very different characteristics and the results from one model cannot be generalized to other models.

Regardless of mutation rates, Figure 7 shows that the final best fitness improves as the mutation rates increase. However, this improvement stops at the mutation rate of 6%, when the final fitness starts to decline as the mutation rates increase. The only exception is Hamming bound 0, which reaches its highest best fitness of 7.62 under a mutation rate of 10%. Table 2 gives the mutation rates used to achieve the best final fitness for all Hamming bounds.

6.2 Methods of Analysis

With three inputs and one output, there are 256 possible Boolean parity functions [16, p. 216]. Each of the 256 functions is rewarded with a different fitness value based on the number of test cases it handles correctly. Figure 8 gives the fitness distribution of the 256 functions in the search space.

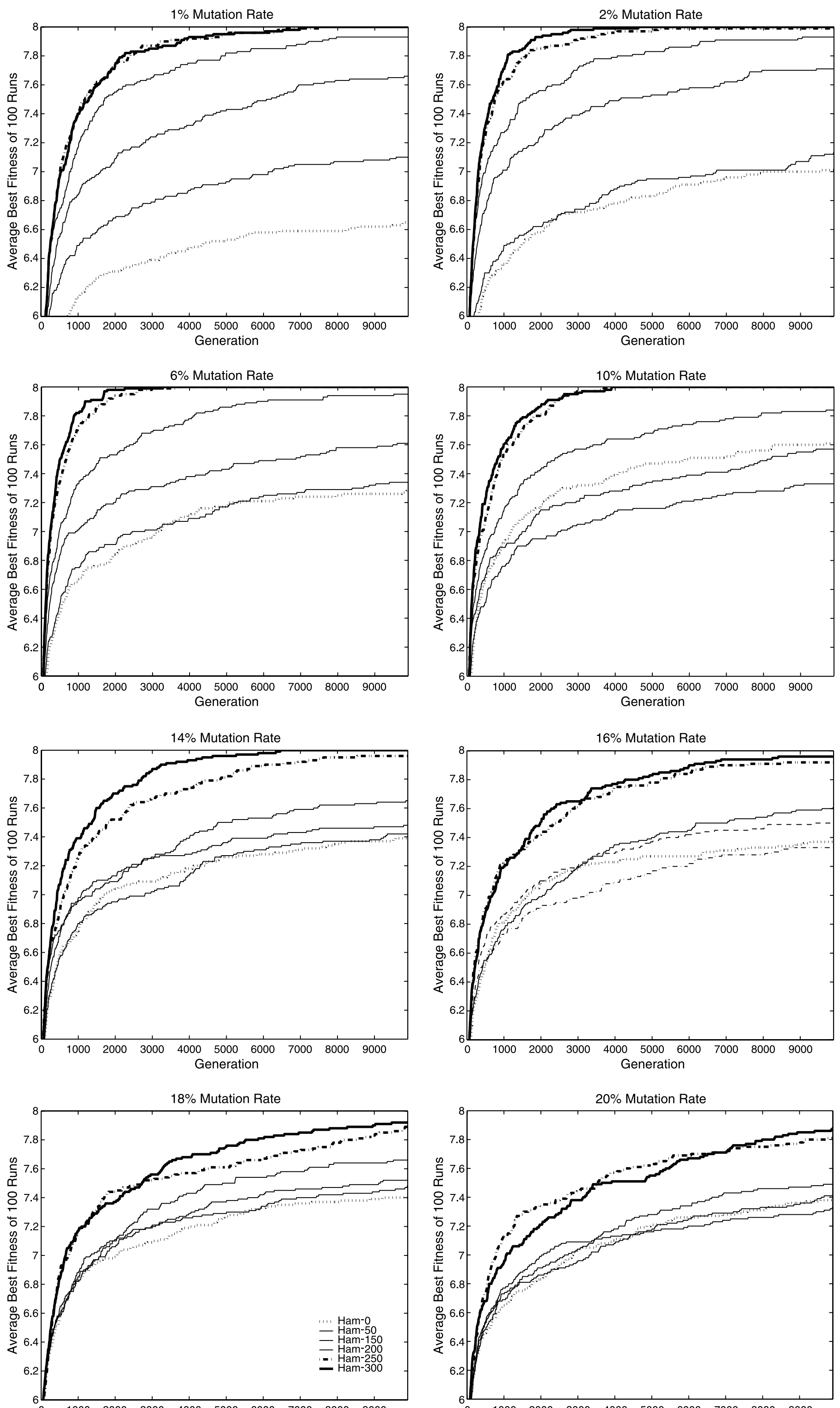

Figure 7. Best fitness with different neutrality levels and mutation rates.

Table 2. Best final fitness for different Hamming bounds.

Best final fitness	Hamming bound	Mutation rate (%)
7.62	Ham-0	10
7.54	Ham-50	12
7.72	Ham-150	2
7.95	Ham-200	6
8.0	Ham-250	1, 2, 4, 6, 8, 10, 12
8.0	Ham-300	1, 2, 4, 6, 8, 10, 12, 14

Since there are many more functions with fitness 4, random search is more likely to generate a function with fitness 4.

After the initial population, evolutionary search starts the process of neutral walk to find genotypes with better fitness. The alternation of neutral walk and fitness improvement steps continues until a fitness 8 genotype is reached (see Figure 8). We analyze the search performance—defined as the ratio of active to inactive gene changes during neutral walk and for fitness improvement—in each of three subspaces (fitness 5 and 6, fitness 6 and 7, and fitness 7 and 8. The calculation of this ratio is demonstrated in Figure 9: G9 is the *first* winner with fitness 5, G10 is the *last* winner with fitness 5, and G11 is the *first* winner with fitness 6. Between G9 and G10, the number of active gene changes is 9 and the number of inactive gene changes is 2. The ratio for neutral walk is 9/2. Between G9 and G11, the numbers of active and inactive gene changes are 10 and 2. The ratio for fitness improvement is 10/2.

We also calculate the percentage of the 100 runs that successfully reached the fitness improvement. We only use data on runs that have sequential fitness improvement (e.g., 4 to 5 to 6). This is because runs that had winning genotypes whose fitness was improved in a non-sequential manner

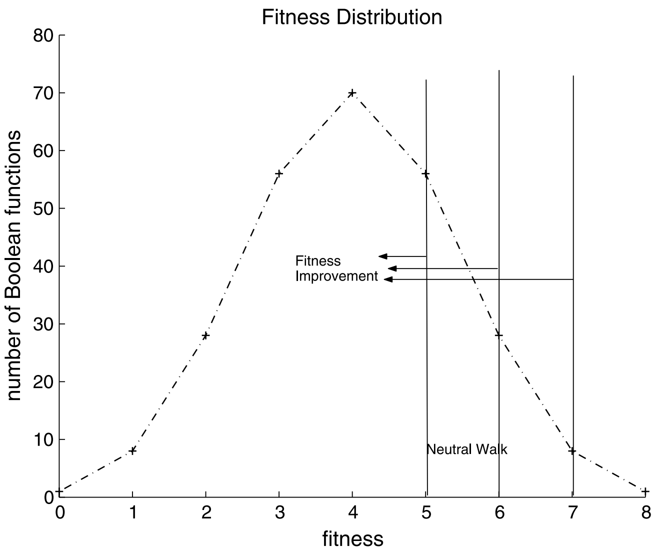

Figure 8. Fitness distribution in the even-3-parity search space.

```
Node output link     5     6     7     8     9
   G9:              4 2 0 0 1 1 1 3 0 5 7 1 8 5 1
   G10:             1 4 0 0 3 0 5 3 1 6 0 0 7 3 1
                    .............................
                    1 1     1 1   1 1 1 1 1 1 1
     Active/Inactive gene change ratio during neutral walk: 9/2
   G9:              4 2 0 0 1 1 1 3 0 5 7 1 8 5 1
   G10:             2 4 1 0 3 0 5 4 1 5 0 0 7 8 1
                    .............................
                    1 1 1   1 1 1 1 1   1 1 1 1
     Active/Inactive gene change ratio for fitness improvement: 10/2
```

Figure 9. Calculating ratios.

(e.g., 4 to 6 to 7 to 8) were quite rare, and they complicate the analysis of the search characteristics. The following subsections analyze the success rates, the ratio of active to inactive gene changes, and their relationship.

6.3 Analysis of Success Rates

The success rates are correlated with the Hamming level: the higher the Hamming level is, the higher the success rates (see Figure 10). Yet, beyond Hamming bound 250, success rates become stable. No significant advantage was observed by adding more neutrality. The success rates are also correlated with mutation rates: with the increase of mutation rates, the success rates also increase. However, when the fitness is close to the optimum, higher mutation rates give lower success rates (see the fitness 7 and 8 search space). This indicates that mutation is destructive in transmitting the fitness of an individual to a fitter one when the fitness is close to optimum. This result is consistent with that reported by other researchers about the difficulty of conserving fitness scores when the population has become fit [1]. These two observations strongly indicate that high success rates are connected with high neutrality in its ability to defend against destructive mutations (in the high fitness region).

The combination of different Hamming levels and mutation rates also reveals interesting success rate behavior. When the fitness is 5 or 6, low Hamming bounds with low mutation rates give low success rates. This can be explained by the fact that a low Hamming bound allows only a small degree of genetic drift. A mutation operation has to transform a genotype from lower fitness into higher fitness using a small number of mutation steps. With low mutation rate, this can be quite difficult to achieve; hence the success rate is low. To achieve high success rates, either a higher Hamming bound or a higher mutation rate is needed. This pattern changes when fitness is 7: only a high Hamming bound with a low mutation rate gives high success rates. All other combinations fail to achieve the optimum of 8.

6.4 Analysis of the Ratio of Active to Inactive Gene Changes

The ratio of active to inactive gene changes increases as the mutation rate increases. This indicates mutation rates have more effect on active gene changes. In other words, mutations are associated with exploitation during the search. In contrast, the ratio decreases as the Hamming bound increases. This indicates Hamming bounds have more effect on inactive gene changes, that is, neutrality promotes exploration.

The ratios for the fitness 5 to 6 search space (between 0.78 and 0.34) are higher than those for the fitness 6 to 7 search space (between 0.65 and 0.3), which are higher than those of the fitness 7 to 8 search space (between 0.5 and 0.2). This means that at the beginning of the evolutionary search, there were more active gene changes, and at the end there were more inactive gene changes. This makes sense, since at the beginning of the search the best fitness of a population is generally low. Active gene changes can easily produce a genotype with improved fitness, and this ends the neutral walk. However, once the genotypes become fitter, most active gene changes lead to a worse genotype, hence leaving the current winner intact. Consequently, the search undergoes neutral walk

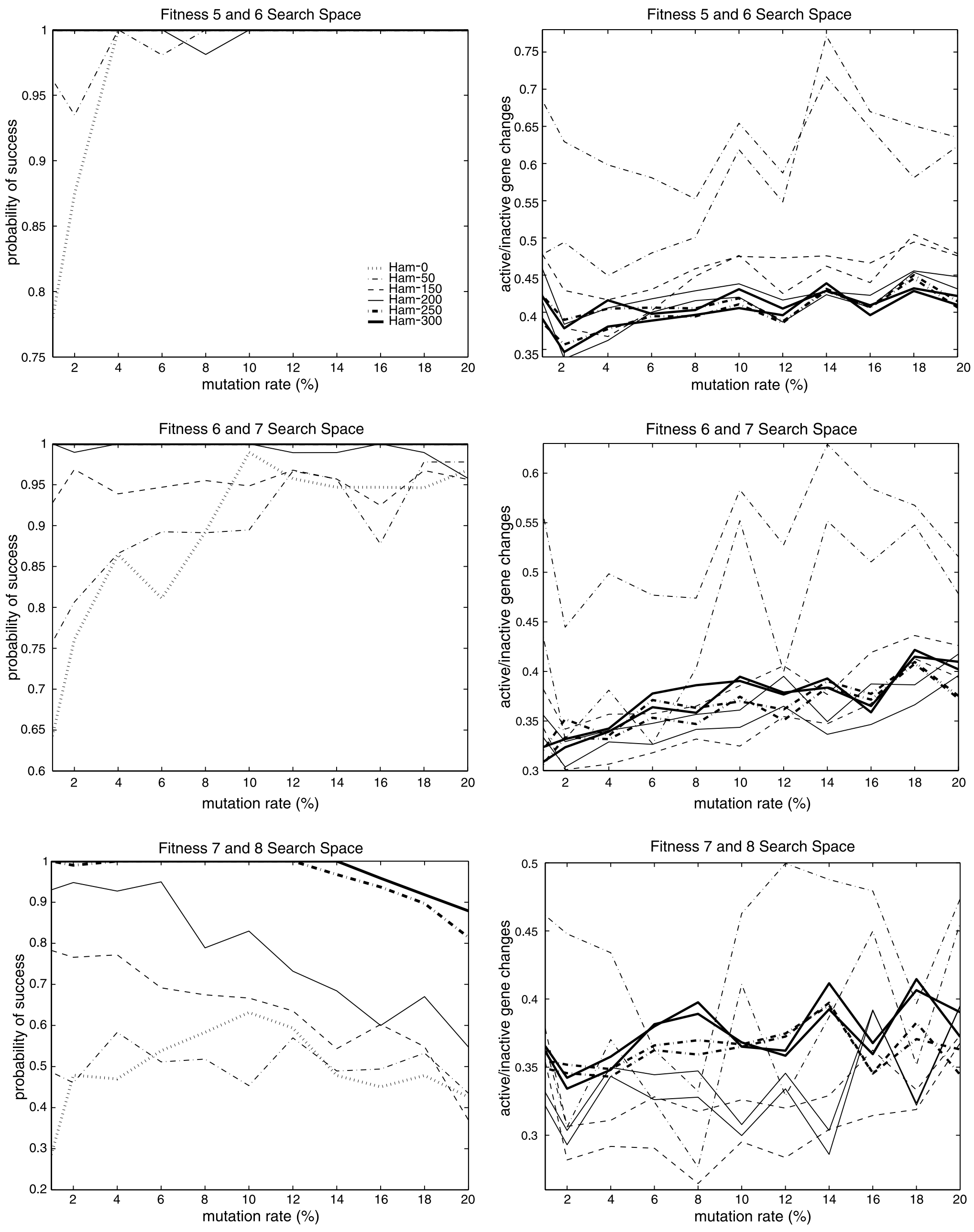

Figure 10. Success rates versus active/inactive gene change ratio for Boolean functions.

with mostly inactive gene changes. It is only when sufficient exploration has obtained the desired genes that the switch of inactive genes to active (counted as active gene changes) leads to a genotype with improved fitness and ends the neutral walk. This view of the search process is confirmed by the data, which shows the ratio for fitness improvement is indeed higher than that of neutral walk.

If, at the beginning of the search, active gene changes lead to fitness improvement more easily than that at the end, the number of generations it takes to find fitter genotypes should be smaller at the beginning of the search. The data plotted in Figure 11 supports this proposition. The plots also

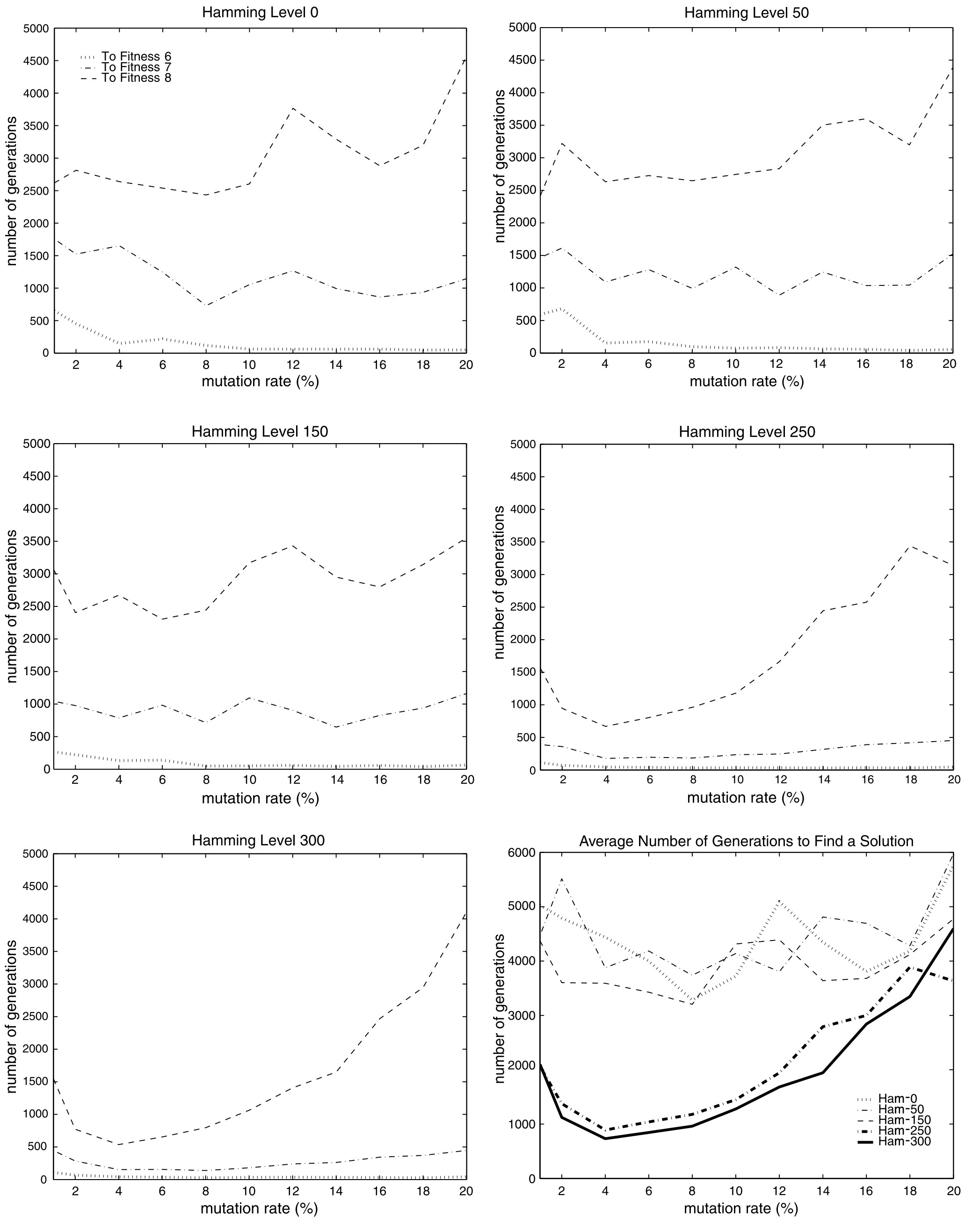

Figure 11. Number of generations required to reach fitness improvement.

show that for the final fitness improvement step (7 to 8), lower mutation rates (between 4% and 8%) give the best results. In contrast, for earlier fitness improvement (5 to 6 and 6 to 7), the average number of generations is only weakly dependent on mutation rates. We therefore conclude that the average time to find optimum fitness is more dependent on the neutrality level (the length of the allowed neutral walks) than on mutation rates. A small Hamming bound with a high mutation rate will take a longer time to reach the optimum than a high Hamming bound with a small mutation rate. Indeed, the total number of generations taken to find a solution, plotted in the bottom right corner of Figure 11, agrees with this conclusion. This contrasts with the findings of Knowles and Watson, who reported that high neutrality led to slower convergence than non-neutrality with a high mutation rate.

6.5 Relationship between the Ratio and High Success Rate

With the information on success rates and the ratios of active to inactive gene changes, we can now identify what type of ratio variations gives high success rates. The most obvious one is that the ratio for neutral walk is close to that of fitness improvement. For example, in the fitness 5 and 6 search space, Hamming bound 300 with mutation rate 10% generates an exploitation/exploration ratio of 0.43 at the fitness improvement step, that is, for a genotype of fitness 5 to transform to become a genotype of fitness 6, the number of active gene changes versus the number of inactive changes is 0.43. The ratio provided by neutral walk is 0.40, which is very close to what is required for successful transformation. This suggests that neutral walk has led the evolutionary search toward the fitness improvement step, and hence enabled the genotype transformation to take place. We also examine the two combinations that have lower success rates (Hamming bound 50 with mutation rates 1% and 2%). In both cases, the two ratios are significantly different from each other. This suggests that neutral walk did not provide satisfactory exploitation/exploration; hence the evolutionary search fails to reach the fitness improvement step.

However, a smaller ratio difference does not always correspond to a higher success rate. For example, in the fitness 6 and 7 search space, Hamming bound 50 with mutation rate 10% has a lower ratio difference than with mutation rate 12% (0.03 vs. 0.127). Yet, it has a lower success rate (89% vs. 96%). We suspect this is due to the ruggedness of the landscape (suggested by the shape of the ratio curves). Smooth ratio curves are the second characteristic we can identify with a high success rate.

In the fitness 7 and 8 search space, Hamming bound 200 also has a very small ratio difference (0.009). However, the success rates are considerably lower than for Hamming bounds 250 and 300. We suspect 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 magnitudes of the exploitation/exploration ratios of those with 100% success rates and found that they are all between 0.34 and 0.41. On the contrary, the Hamming bound 200 gives higher inactive gene changes, that is, more exploration. This might lead the evolution toward the areas where convergence is harder.

Based on this analysis, we have identified the following characteristics of exploitation/exploration ratios that are associated with high success rates:

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

Both Hamming bounds 250 and 300 give high success rates in all three subspaces. Yet they all have different exploitation/exploration ratios. As discussed in the previous section, successful search requires the ratio to be higher at the beginning (fitness 5 and 6 search space) and lower at the end (fitness 7 and 8 search space). Recall also that mutations assist exploitation, and to reduce the amount of exploitation at the end of the search process, a system without neutrality would need to change the parameter setting or use a self-adaptive mutation operator. However, in our system, no parameter or self-adaptive operator is employed. Yet, the ratio adjustment takes place on its own. How did this happen? The credit belongs to the type of genetic redundancy that we employ.

At the beginning of the search process, mutations can easily find a fitter genotype. Although a large amount of freedom is given to the neutral walk, only a few neutral steps are needed (between 27 and 108, depending on mutation rates). As the evolutionary search proceeds, more neutral steps were taken advantage of: between 139 and 446 to find a fitness 7 genotype, and between 537 and

4107 to find a fitness 8 genotype. Recall that neutrality promotes exploration. The incremental increase of the neutral walk caused the gradual reduction of the exploitation/exploration ratio and led to a successful search.

7 Experiments on Needle-in-Haystack Fitness Landscapes

The second set of experiments was on a landscape where only two fitness values are possible: figuratively, the needle and the hay. In general, heuristic search algorithms cannot improve on the performance of random search algorithms, since there is no fitness information that suggests which way to go. We chose to investigate this problem because it is very different from a situation where there is a fitness gradient and we felt it would be illuminating to see how neutrality affects the search in such a landscape. To investigate how the presence of neutral networks influences evolutionary search, we modified the previously described even-parity functions to have a needle-in-haystack property by changing the function set to contain only one Boolean function, EQ. With this, every evolved program either gets half of the test cases right or gets all of them right [18, 42].

Note that the even-parity problem has to have an even number of inputs (e.g., 4, 6, 8,...) in order to be constructable from EQ alone. For example, with even-8-parity, the number of Boolean inputs is eight. There are 2^8 different possible input combinations, hence 256 test cases. An even-8-parity constructed using EQ would have a fitness of either 128 or 256.

Table 3 shows the three problems we investigated in this work. Note that the random search success rates were calculated by randomly generating 1,000,000 programs, except for even-12-parity, where 4,000,000 trials were made (since it's a more difficult problem). No solution with fitness 4096 was found in this set of trials.

When random genotypes are generated, one finds that the number of non-solutions vastly outnumbers the number of solutions. This is caused by the fact that many genotypes code for a phenotype that does not use all the required inputs. We do not constrain the genotypes so that all inputs must be used.

For evolutionary search experiments, the setups are mostly the same as that of the Boolean function landscape problem (see Table 1 for details). The only difference is that we included one more neutrality level (Hamming bound 100) for more detailed analysis.

7.1 Results

We do not present the results on best fitness, because they do not give performance information for needle-in-haystack problems, where only two fitness values are possible. For example, with even-8-parity, the best fitness in the initial population is 128. This value remains the best fitness until a

Table 3. Even-parity problems with needle-in-haystack fitness landscape.

Problem	Even-8-Parity	Even-10-Parity	Even-12-Parity
Function set	EQ	EQ	EQ
Terminal set	x_1 to x_8	x_1 to x_{10}	x_1 to x_{12}
No. of test cases	256	1024	4096
Possible fitness	128, 256	512, 1024	2048, 4096
Random search success rate	0.063%	0.0054%	No solution found

solution with fitness 256 is found. This behavior is the same for all neutrality levels and mutation rates, and hence is not useful for identifying the effect of neutrality on the search performance. Instead, we evaluate the performance based on the success rates.

Figure 12 shows that the combination of low Hamming bound and low mutation rate gives low success rates for even-8-parity. When the mutation rate is 1% or 2%, a small amount of neutrality

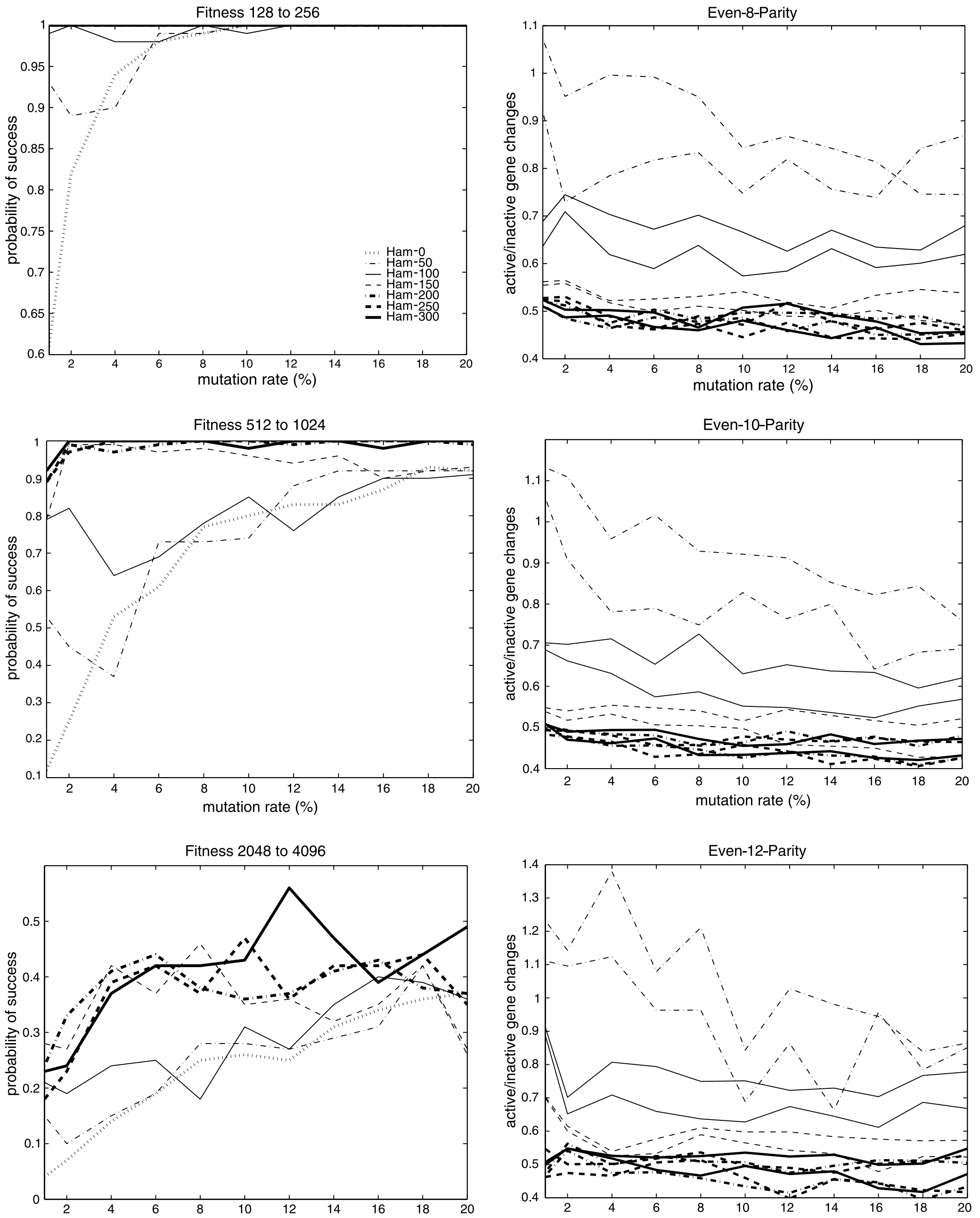

Figure 12. Success rates versus active/inactive gene change ratio for needle-in-haystack problem.

(50) is able to improve success rates. However, when the mutation rate is 4%, a neutrality level equal to Hamming distance 100 is required to improve the performance. For example, increasing the mutation rate from 2% to 4% gives Hamming bound 0 a success rate jump from 82% to 94%. In comparison, increasing the neutrality level from 0 to 50 at a mutation rate of 4% does not improve the success rate. This suggests that raising the mutation rate has a stronger impact than raising the neutrality level on the evolutionary search. The relationship between neutrality and mutation rates will be discussed in more detail in Section 7.5.

Regardless of mutation rate, increasing the neutrality level beyond 150 does not improve or impair the performance; all of them have 100% success rate. This suggests that the equilibrium between the benefits of exploitation and exploration is reached at this point. Indeed, the analysis of active/inactive gene change ratios shows that the exploitation/exploration ratios are very similar for all Hamming bounds beyond 150 (see Section 7.3). Increasing neutrality or mutation rates do not affect this equilibrium.

Even-10-parity is harder than even-8-parity: there are more unsuccessful runs, especially when low Hamming bounds were used. The behavior is similar to that of even-8-parity: the success rate remains approximately the same after a certain Hamming bound is reached (200 is the equilibrium point). Moreover, mutation rates are more influential in this problem: neutrality level 150 is required to give consistent improvement of success rates (in contrast to neutrality level 100 in even-8-parity). This indicates that when the problem becomes more difficult, the role of neutral walks in increasing the success rate becomes more evident.

Even-12-parity is the hardest among all; none of the implementations has 100% success rate. Although it is not as clear as for the other problems, even-12-parity reaches the equilibrium point around neutrality level 200 with success rates around 40%. This indicates that high neutrality and mutation rate are not sufficient for the search algorithm to find a solution to this problem. Modification of other parameters, such as genotype length and the maximum number of generation, will be required to improve performance.

With respect to performance, the seven different Hamming bound implementations can be roughly divided into two groups: the first group consists of neutrality levels 0, 50, and 100, and the second group consists of neutrality levels 150, 200, 250, and 300. In the first group, increasing mutation rates increases success rates, while in the second group little performance improvement is gained after the mutation rate exceeds 4%. However that behavior changes in the hardest problem (even-12-parity). Here, although the success rates are lower for all scenarios, we begin to see that higher mutation is of benefit at all neutrality levels.

7.2 Analysis of Success Rates

Examining the experimental results for all three needle-in-haystack problems, we have made the following observations:

- Success rates improve as the neutrality level increases.
- However, beyond a certain neutrality level, the success rates become stable.
- Increasing mutation rates also improves success rates.
- For the even-12-parity problem, which is more difficult than the others, a high neutrality level and mutation rate are not sufficient for the evolutionary algorithm to achieve a 100% success rate.

These results are very similar to the results for even-3-parity in the previous section. The only difference is that higher mutation rates *improve* success rates in needle-in-haystack landscapes, whereas they *impair* the performance of even-3-parity when the population becomes fit. The results from this set of experiments also make it clear that there exists an equilibrium point, which gives the exploitation/exploration balance corresponding to the maximum possible performance. Increasing neutrality or mutation rates beyond this point has little effect on the performance. For different problems, there is a different equilibrium point: Hamming bound 150 for even-8-parity, Hamming bound 200 for both even-10-parity and even-12-parity, and Hamming bound 250 for even-3-parity in the previous section.

Another similarity is that a high Hamming bound also requires less generations to find a solution (see Figure 13). This applies to both even-8-parity and even-10-parity. Note that we did not plot the results for Hamming bounds 0 and 50 under mutation rates 1%, 2%, and 4% for even-10-parity. This is because the number of successful runs is too low to be statistically meaningful. The same applies to even-12-parity, where no implementation has a success rate more than 58%. We also examined the trend of these plots and found that the best performance (smallest number of generations) occurs when both Hamming bound and mutation rate are high. This is different from the even-3-parity in the previous section, where the fitness landscape is not needle-in-haystack. In that experiment, the best performance occurs when the Hamming bound is high but the mutation rate is low (4%) (see Figure 11).

7.3 Analysis of the Ratio of Active to Inactive Gene Changes

The ratios of active to inactive gene changes decrease as mutation rates increase. Similarly, the ratios also decrease as Hamming bounds increase. This indicates that both mutation and neutrality have more effect on inactive gene changes during evolutionary search. In other words, they both promote *exploration*. This behavior differs from that found with even-3-parity, where mutations play the role of exploitation while neutrality plays the role of exploration.

However, similar to the those found for even-3-parity, the ratio for fitness improvement is higher than that for neutral walk. Thus, what we perceived to be happening during evolutionary search of even-3-parity can also be applied to needle-in-haystack landscapes: neutral walks mostly explore inactive gene changes, and when the desired genes were obtained, the switch of inactive genes to active (counted as active gene changes) leads to a genotype with improved fitness.

7.4 Relationship between the Ratio and High Success Rate

Again the results are similar to those of even-3-parity in the previous section: high success rates are associated with a small ratio gap between neutral walk and fitness improvement. For example, all the Hamming bounds 150, 200, 250, and 300 gave 100% success rate for even-8-parity. The corresponding ratio gaps between neutral walk and fitness improvement are in the vicinities of 0.6 and 0.4. In contrast, combinations produce large ratio gaps (Hamming bound 50 with mutation rates 1%, 2%, and 4%) have low success rates. A similar pattern is also observed in even-10-parity and even-12-parity results.

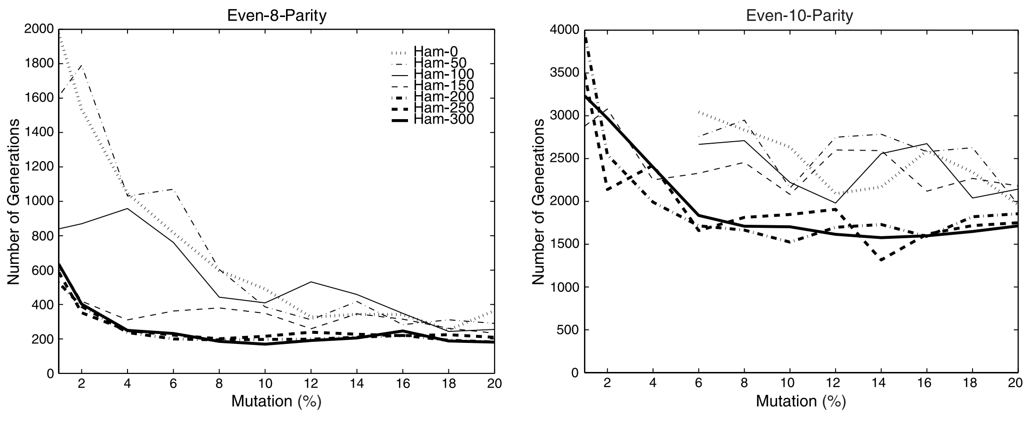

Figure 13. Number of generations required to find a solution.

The range of ratios for high success rates is between 0.75 and 0.43 for even-8-parity and between 0.55 and 0.40 for even-10-parity. In contrast, even-12-parity has a ratio pattern showing higher active gene changes, which are also associated with lower success rates. This suggests that the algorithm does not provide sufficient inactive gene change (exploration) for the search to find a solution in even-12-parity problem. Overall, each time the problem difficulty increases (from even-5 to even-8 to even-10 to even-12), more inactive gene changes were required for the evolutionary search to be successful. This suggests exploration is more important than exploitation for search in a needle-in-haystack type of space. This is exactly what one would expect for such a landscape.

7.5 Relationship between Neutrality and Mutation Rates

Experimental results from three needle-in-haystack problems show that both neutrality and mutation rates have an inverse relationship with the active/inactive gene change ratio: the higher they are, the lower the ratio. In other words, increasing neutrality or mutation rates increases inactive gene changes. Such increase of exploration also improves success rates.

However, this relationship does not apply to search space that has more than two possible fitness values. In the previous section, even-3-parity has eight possible fitness values. The study indicates that higher mutation rates give higher active gene changes. Such an increase of exploitation is beneficial when population average fitness is low, but detrimental when the population has become fit.

We compared the success rates of even-8-parity (Figure 12) and that for even-3-parity fitness 5 to 6 search space (Figure 10) and found them to have similar behavior. This suggests the fitness 5 to 6 search space has the needle-in-haystack property but with a large number of needles (fitness 6 genotypes). Both mutation and neutrality have positive correlation with the success rates in this subspace.

Since neutrality and mutation rates have a similar search effects in needle-in-haystack problems, can they replace each other? We believe they complement each other in this type of problem. In other words, a low mutation rate with a high Hamming bound gives similar results to a high mutation rate with a low Hamming bound. However, the following has to be noted:

- There is no quantitative indication of which neutrality level gives the same performance as that corresponding to a given mutation rate. For example, 50 neutral mutations are about 16% of the total number of 300 genes. Yet, neutrality level 50 with 1% mutation rate does not give the same performance as neutrality level 0 with 16% mutation rate.
- Once an equilibrium point is reached, the success rate is approximately the same regardless of neutrality level and mutation rate, that is, the relation between neutrality and mutation rate is irrelevant to the search performance.

8 Discussion

Two different types of problems were investigated using a Wright-inspired neutrality system. Although their search spaces have different characteristics, evolutionary search was able to find better solutions faster when a higher neutrality level was employed. Through a novel analysis based on the ratio of exploitation and exploration, we have found that different ratios are required in different landscapes to ensure successful search: Boolean landscapes require an increased ratio when a higher mutation rate is used, while needle-in-haystack landscapes require a decreased ratio.

Moreover, this ratio changes as the search progresses from one part of the search space to another: the ratio is higher in early stages than that in later stages of the search in the Boolean even-3-parity problem. In any event, the dynamics is well handled by the neutrality system, according to our analysis of the data. We believe the neutrality model we have presented here, based on the interaction of active and interactive genes, has made an original contribution to the study of neutrality in EC systems.

In the presence of a neutral network, evolutionary search is permitted to explore implicit neutrality (through active genes changes) and explicit neutrality (through inactive gene changes). Moreover, the structure of the neutral network is not fixed, but changes according to the interaction of active and inactive genes. This interaction is the engine that makes the adjustment of the exploitation/exploration ratio. It is essential to the success of an evolutionary search.

The ratio study also leads to an insight of how the evolutionary search is carried out under the presence of this kind of neutrality:

Evolutionary search is a sequence of neutral walk and fitness improvement steps. During neutral walk, mutations exploit implicit neutrality and explore explicit neutrality. When the ratio of exploitation and exploration is close to an optimum level, evolutionary search finds a fitter genotype.

This hypothesis has been tested against two types of problems and has gained support in both cases. We will investigate other types of problems to confirm or reject this hypothesis.

As with other concepts in biology (e.g., selection and mutation), neutrality can be implemented in many different ways. We have reviewed those devised by others and found each has different characteristics (see Section 5). Therefore, the results from one model cannot easily be generalized to others. One current model that has been strongly debated is the many-to-one genotype-phenotype mapping method by Ebner and colleagues [6]. In addition to Knowles and Watson [15], Bullock [4] pointed out that what is important is not just the *number* of phenotypes that neighbor a neutral network, but the *frequency distribution* over this neighborhood, and where it is significantly biased.

There have been suggestions that there must be "no free lunch for neutrality," since it is easy to argue that no technique is superior over all problems. Although we consider the form of neutrality that we have implemented unique and have shown that it leads to a more effective evolutionary search, we look forward other researchers examinations so that greater understanding may be gained regarding its usefulness.

9 Conclusions

Neutrality is a misunderstood subject in the EC community. This is due partly to its subtlety, partly to its novelty, and sometimes to the lack of a clear or comprehensive discussion of the literature. As a result, we see both positive and negative publications on this subject. On the one hand, this provokes interest from the research community and brings new contributions to this subject. On the other hand, it also can discourage people from working on the subject if they are misled by the literature. We hope it is the former that is happening.

In this article, we have presented a detailed introduction to neutrality, including the distinction of two types of neutrality (implicit and explicit); compared and contrasted different neutrality models; introduced a neutrality model based on the interaction of neutral and adaptive mutations; and given a detailed analysis of its search properties.

For the two types of problems we have investigated, neutrality is beneficial. The higher the allowed neutral walk length, the better the final solution is. We also identify that the advantage comes from the interaction of adaptive and neutral mutations, which automatically adjust the exploitation and exploration for the evolutionary search to find fitter solutions.

Although it has given many promising results, this type of neutrality still has many open issues that require further investigation. For example, does the exploitation/exploration ratio of a problem depend on the length of the genotype? Is the ratio the right measure to use? Also, for the two types of landscapes, Boolean function problems are sensitive to mutation rates while needle-in-haystack problems are not. What will happen in other types of landscapes? Can we formulate a generalized view of mutation rates under neutrality? We continue this research to better understand the characteristics of this type of neutrality.

References

1. Altenberg, L. (1994). The evolution of evolvability in genetic programming. In K. E. Kinner, Jr. (Ed.), *Advances in genetic programming* (pp. 47–74). Cambridge, MA: MIT Press.
2. Barnett, L. (2001). Netcrawling-optimal evolutionary search with neutral networks. *Proceedings of the 2001 Congress on Evolutionary Computation* (pp. 3037–3045). Piscataway, NJ: IEEE Press.
3. Barnett, L. (2002). *Evolutionary search on fitness landscapes with neutral networks.* Unpublished doctoral dissertation, CCNR and COGS, University of Sussex.
4. Bullock, S. (2001). Smooth operator? Understanding and visualising mutation bias. *Proceedings of the 6th European Conference on Artificial Life* (pp. 602–612). Berlin: Springer-Verlag.
5. Davidson, E. H. (2001). *Genomic regulatory systems: Development and evolution*. San Diego, CA: Academic Press.
6. Ebner, M., Shackleton, M., & Shipman, R. (2001). How neutral networks influence evolvability. *Complexity*, *7*(2), 19–33.
7. Ebner, M., Langguth, P., Albert, J., Shackleton, M., & Shipman, R. (2001). On neutral networks and evolvability. *Proceedings of the 2001 Congress on Evolutionary Computation* (pp. 1–8). Piscataway, NJ: IEEE Press.
8. Forrest, S., & Mitchell, M. (1992). Relative building-block fitness and the building-block hypothesis. *Foundations of Genetic Algorithms 2* (pp. 109–126). San Mateo, CA: Morgan Kaufmann.
9. Harik, G. R., & Goldberg, D. E. (1996). Learning linkage. *Foundations of Genetic Algorithms 4* (pp. 247–262). San Mateo, CA: Morgan Kaufmann.
10. Harvey, I., & Thompson, A. (1996). Through the labyrinth evolution finds a way: A silicon ridge. *Proceedings of the First International Conference on Evolvable Systems: From Biology to Hardware* (pp. 406–422). Berlin: Springer-Verlag.
11. Harvey, I. (1992). Species adaptation genetic algorithms: A basis for a continuing SAGA. *Proceedings of the First European Conference on Artificial Life* (pp. 346–354). Cambridge, MA: MIT Press.
12. Huynen, M. A., Stadler, P. F., & Fontana, W. (1996). Smoothness within ruggedness: The role of neutrality in adaptation. *Proceedings of the National Academy of Sciences, U.S.A.*, *93*, 397–401.
13. Kimura, K. (1969). Evolutionary rate at the molecular level. *Nature*, *217*, 624–626.
14. King, J. L., & Jukes, T. H. (1969). Non-Darwinian evolution. *Science*, *164*, 788–798.
15. Knowles, D., & Watson, R. A. (2002). On the utility of redundancy encodings in mutation-based evolutionary search. *Parallel Problem Solving from Nature—PPSN VII Seventh International Conference* (pp. 88–98). Berlin: Springer-Verlag.
16. Koza, J. R. (1992). *Genetic programming: On the programming of computers by means of natural selection.* Cambridge, MA: MIT Press.
17. Kreitman, M. (1996). The neutral theory is dead. Long live the neutral theory. *BioEssays*, *18*, 678–682.
18. Langdon, W. B., & Poli, R. (1998). *Why "building blocks" don't work on parity problems* (Technical report CSRP-98-17). The University of Birmingham.
19. 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* (pp. 123–127). San Mateo, CA: Morgan Kaufmann.
20. Miller, J. F., & Thomson, P. (2000). Cartesian genetic programming. *Proceedings of the Third European Conference on Genetic Programming* (pp. 121–132). Berlin: Springer-Verlag.
21. Miller, J. F., Thomson, P., & Fogarty, T. C. (1997). Designing electronic circuits using evolutionary algorithms. Arithmetic circuits: A case study. In D. Quagliarella, J. Periaux, C. Poloni, & J. Winter (Eds.), *Genetic algorithms and evolution strategies in engineering and computer science* (pp. 105–131). New York: Wiley.
22. Miller, J. F., Vassilev, V. K., & Job, D. (2000). Principles in the evolutionary design of digital circuits—Part I. *Genetic Programming and Evolvable Machines*, *1*(1/2), 7–35.
23. Garmendia-Doval, B. A., Miller, J. F., & Morley, S. D. (2004). Cartesian genetic programming and the post docking filtering problem. In U.-M. O'Reilly, T. Yu, R. Riolo, & B. Worzel (Eds.), *Genetic programming theory and practice II* (pp. 225–244). Dordrecht, The Netherlands: Kluwer.

24. Newman, M. E. J., & Engelhardt, R. (1998). Effects of neutral selection on the evolution of molecular species. *Proceedings of the Royal Society of London Series B*, *265*, 1333–1338.

25. Ohta, T. (1976). Role of very slight deleterious mutations in molecular evolution and polymorphism. *Theoretical Population Biology*, *10*, 254–275.

26. Ohta, T. (1977). Extension to the neutral mutation random drift hypothesis. *Proceedings of the Second Taniguchi International Symposium on Biophysics* (pp. 148–167).

27. Ohta, T. (1992). The nearly neutral theory of molecular evolution. *Annual Reviews of Ecology & Systematics*, *23*, 263–286.

28. McDonald, J. H., & Kreitman, M. (1991). Adaptive protein evolution at the adh locus in *Drosophila*. *Nature*, *351*, 652–654.

29. Poli, R. (1996). *Parallel distributed genetic programming* (Technical report CSRP-96-15). University of Birmingham, UK.

30. Provine, W. B. (1986). *Sewall Wright and evolutionary biology*. Chicago: The University of Chicago Press.

31. Rothlauf, F., & Goldberg, D. E. (2003). Redundant representations in evolutionary computation. *Evolutionary Computation*, *11*(4), 381–415.

32. Schwefel, H. P. (1965). *Kybernetische Evolution als Strategie der experimentellen Forschung in der Stromungstechnik*. Diplomarbeit, Technische Universität Berlin.

33. Wilke, C. O., Wang, J. L., Ofria, C., Lenski, R. E., & Adami, C. (2001). Evolution of digital organisms at high mutation rate leads to survival of the flattest. *Nature*, *412*, 331–333.

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

35. van Nimwegen, E., & Crutchfield, J. P. (2001). Optimizing epochal evolutionary search: Population-size dependent theory. *Machine Learning*, *45*, 77–114.

36. van Nimwegen, E., Crutchfield, J. P., & Huynen, M. (1999). *Neutral evolution of mutational robustness* (Technical report 99-03-021). Santa Fe, NM: Santa Fe Institute.

37. Vassilev, V. K., & Miller, J. F. (2000). The advantages of landscape neutrality in digital circuit evolution. *Proceedings of the Third International Conference on Evolvable Systems: From Biology to Hardware* (pp. 252–263). Berlin: Springer-Verlag.

38. Wagner, A., & Stadler, P. F. (1999). *Viral RNA and evolved mutational robustness* (Technical report 99-02-010). Santa Fe, NM: Santa Fe Institute.

39. Yu, T., & Bentley, P. (1998). Methods to evolve legal phenotypes. *Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature* (pp. 280–291). Berlin: Springer-Verlag.

40. Yu, T., & Miller, J. (2001). Neutrality and the evolvability of Boolean function landscape. *Proceedings of the Fourth European Conference on Genetic Programming* (pp. 204–217). Berlin: Springer-Verlag.

41. Yu, T., & Miller, J. (2002). Finding needles in haystacks is not hard with neutrality. *Proceedings of the Fifth European Conference on Genetic Programming* (pp. 13–25). Berlin: Springer-Verlag.

42. Yu, T. (1999). Structure abstraction and genetic programming. *Proceedings of the 1999 Congress on Evolutionary Computation* (pp. 652–659). Piscataway, NJ: IEEE Press.