On the effects of bit-wise neutrality on fitness distance correlation, phenotypic mutation rates and problem hardness

Riccardo Poli and Edgar Galván-López

Department of Computer Science
University of Essex
Colchester, CO4 3SQ, UK
{rpoli,egalva}@essex.ac.uk

Abstract. The effects of neutrality on evolutionary search are not fully understood. In this paper we make an effort to shed some light on how and why bit-wise neutrality – an important form of neutrality induced by a genotype-phenotype map where each phenotypic bit is obtained by transforming a group of genotypic bits via an encoding function – influences the behaviour of a mutation-based GA on functions of unitation. To do so we study how the fitness distance correlation (*fdc*) of landscapes changes under the effect of different (neutral) encodings. We also study how phenotypic mutation rates change as a function of the genotypic mutation rate for different encodings. This allows us to formulate simple explanations for why the behaviour of a GA changes so radically with different types of neutrality and mutation rates. Finally, we corroborate these conjectures with extensive empirical experimentation.

1 Introduction

Evolutionary Computation (EC) systems are inspired by the theory of natural evolution. The theory argues that through the process of selection, organisms become adapted to their environments and this is the result of accumulative beneficial mutations. However, in the late 1960s, Kimura [22] put forward the theory that the majority of evolutionary changes at molecular level are the result of random fixation of selectively neutral mutations. In other words, the mutations that take place in the evolutionary process are neither advantageous nor disadvantageous to the survival of individuals. Kimura's theory, called neutral theory of molecular evolution, considers a mutation from one gene to another as neutral if this modification does not affect the phenotype.

Within the context of EC, different approaches have been proposed to study neutrality in evolutionary search. Whether or not neutrality helps evolutionary search, however, has not conclusively been established. In the following section, we will present work that shows clearly that the relationship between the genotype space and phenotype space when neutrality is present in the evolutionary search plays a crucial role.

The aims of our work are:

– understanding the relationship between the solution space (represented at phenotype level) and the search space (represented at genotype level) in the presence of neutrality, and, following this analysis,
– identifying under what circumstances neutrality may help to improve performance of evolutionary processes.

The paper is organised as follows. In the next section, previous work on neutrality in EC is summarised. In Section 3, we describe the genotype-phenotype encodings studied in this paper. In Section 4 we review the notion of the fitness distance correlation (*fdc*), introduce our test problems and look at their *fdc* in the absence of neutrality. In Section 5 we study the effects of bitwise neutrality on the difficulty of our test problems, exploring the case of the OneMax problem in particular depth. Section 6 makes the relation between genotypic mutation rates and phenotypic mutation rates explicit. In Sections 7 and 8 we present and discuss the results of experiments with unimodal, multimodal and deceptive landscape problems and draw some conclusions.

2 Previous Work

In biology the effects of neutrality have been extensively discussed in numerous studies (see for example [22, 18, 13]). As a consequence of these studies, there has been a growing interest in using and analysing the effects of neutrality in EC.

For instance, Barnett [3] introduced a new family of *NK* fitness landscapes that has the property of allowing the explicit addition of neutrality in the evolutionary process. He called them *NKp* fitness landscapes. The parameter p is determines the amount of neutrality that is be present during evolution ($p = 0$ corresponds to a normal *NK* landscape while $p = 1$ corresponds to a flat landscape). Barnetts' motivation was to see if the constant innovation property observed in biology [18] was present in this type of neutrality. The author argued that, at least for a mutation-selection algorithm, avoiding to get stuck in local optima can be achieved in the presence of neutral networks.

In a insightful investigation, Weicker and Weicker [36] used two methods to analyse the effects of redundancy: diploid and decoders. For the former method, each individual contains two solutions and an extra bit which is in charge to set the active solution. So, it is clear that the size of the search space for this kind of redundancy has increased dramatically. Moreover, this kind of mapping is homogenous. This is not the case, however, for the decoder method. A decoder is effectively a repair mechanism that maps an invalid genotype (e.g., one that violates some constraints) into a valid one. They investigated the effects of both methods with respect to local optima, finding that local optima in the search space are converted to plateaus. However, this does not mean that this represent an advantage, as the authors pointed out saying: "... redundancy has many facets with various different characteristics. The mapping from those characteristics to the expected performance remains to be done" [36].

Toussaint and Igel [33] pointed out that standard approaches to self-adaptation [10] are a basic and explicit example for the benefit of neutrality. In these approaches the genome is augmented with strategy parameters which typically parameterise the mutation distribution (e.g., the mutation rate). These neutral parts of the genome are co-adapted during evolution to induce better search distributions. Interestingly, theoretical work on the evolution of strategy parameters [4] can so be re-interpreted as theoretical results on the evolution of neutral traits. The point of view developed in [33] conversely suggests that the core aspect of neutrality is that different genomes in a neutral set provide a variety of different mutation distributions from which evolution may select in a self-adaptive way.

This line of thought was further formalised by Toussaint [32]. Given a fixed GP-map one can investigate the varieties of mutation distributions induced by different genomes in a neutral set. If their phenotypic projections (the phenotypic mutation distributions) are constant over each neutral set this is defined as *trivial neutrality*. Toussaint shows that trivial neutrality is a necessary and sufficient condition for compatibility with phenotypic projection of a mutation-selection GA. Intuitively this means that, in the case of trivial neutrality, neutral traits have no effect on phenotypic evolution. I.e., whether one or another representative of a neutral set is present in a population does not influence the evolution of phenotypes. Note that one of the encodings we will investigate (the Parity encoding) is a case of trivial neutrality. This and calculations presented in Section 5 will help us explain the results presented in Section 7. In the case of non-trivial neutrality, different genotypes in a neutral set induce different phenotypic distributions, which implies a selection between equivalent genotypes similarly to the selection of strategy parameters in self-adaptive EAs. Toussaint interprets this as the underlying mechanism of the evolution of genetic representations.

Vassilev and Miller [35] claimed that the presence of neutrality in evolutionary search was useful when they used Cartesian Genetic Programming (CGP) [25] to evolve digital circuits. For their study, the authors considered the well-known three-bit multiplier problem. They focused their attention on the relation between the size and the height of the landscapes plateaus. In their work, Vassilev and Miller suggested that the length of neutral walks will decrease as the best fitness increases. They concluded that neutrality helps to cross wide landscapes areas of low fitness.

Smith *et al.* [29] analysed the effects of the presence of neutral networks on the evolutionary process. They observed how evolvability was affected by the presence of such neutral networks. For their study they used a system with an extremely complex genotype-to-fitness mapping. They concluded that the existence of neutral networks in the search space, which allows the evolutionary process to escape from local optima, does not necessarily provide any advantage. This is because the population does not evolve any faster due to inherent neutrality. In a different piece of work [30], the same authors focused their research on looking at the dynamics of the population rather than looking at just the fitness, and argued that neutrality did not perform a useful role in an evolutionary robotic task.

Ebner *et al.* [9] studied the effects of neutrality on the search space. For this purpose, they separate the search space into phenotypes which belong to different species. In their work, they proposed three different types of encodings which, according to the authors, seem to allow a high degree of connectivity among neutral networks and so, individuals will not have problem discovering other species. From their experiments, they concluded that the higher the degree of redundancy (another term for neutrality) is, the better species are able to adapt. In other words, redundancy avoids getting stuck in local optima.

Yu and Miller [37] showed in their work that neutrality improves the evolutionary search process for a Boolean benchmark problem. They used Miller's CGP [25] to measure explicit neutrality in the evolutionary process. They explained that mutation on a genotype that has part of its genes active and others inactive may produce different effects: mutation on active genes is adaptive because it exploits accumulated beneficial mutations, while mutation on inactive genes has a neutral effect on a genotype's fitness, yet it provides exploratory power by maintaining genetic diversity. Yu and Miller extended this work in [38] showing that neutrality was helpful and that there is a relationship between neutral mutations and success rate in a Boolean function induction problem. However, Collins [7] claimed that the conclusion that neutrality is beneficial in this problem is flawed.

Yu and Miller also investigated neutrality using the simple OneMax problem [39]. They attempted a theoretical approach in this work. With their experiments, they showed that neutrality is advantageous because it provides a buffer to absorb destructive mutations.

Chow [5] studied the relationship between genotype space and phenotype space. In his work, Chow used a hybrid algorithm (a GA receiving feedback from a hill climber). The approach proposed by Chow relies on replacing a genotype by converting a phenotype to its corresponding genotype. Such phenotype is given to the GA by the hill climber. Chow claimed that in all experiments, such replacements improved the search results.

Fonseca and Correia [12] developed a mathematical model which is able to include the properties proposed by Rothlauf and Goldberg [28] and which are claimed to influence the quality of redundant representations. All their experiments were carried out in the context of a simple mutation-selection evolutionary model. Under this model, the authors were wondering whether a redundant representation might be constructed which preserves evolutionary behaviour. Based on their mathematical model, they claimed that the presence of non-coding genes do not affect the evolutionary process. However, they were unable to determine what kind of representation (redundancy) is necessary to obtain good results on a given optimisation problem.

Banzhaf and Leier [2] studied the effects of neutral networks' connectivity using a Boolean problem. They studied the effects of neutrality using 2 NAND space and showed how it can aid the evolutionary search. For this purpose, Banzhaf and Leier used a linear GP representation because, they argued, with GP it is easier to identify neutrality than in other evolutionary method. In their experiments and by means of an exhaustive examination of all possible genotypes, they showed how there are highly common phenotypes and very few uncommon phenotypes. The authors concluded that neutral networks must be highly intertwined to allow a quick transition from one network to the next.

In summary, the literature presents a mixed picture as to what the effects of neutrality on evolutionary search are.

As can be seen from previous paragraphs, the relationship between phenotype and genotype space is crucial to understand the influence of neutrality on evolutionary search. We believe that the effects of neutrality on evolutionary search are not well understood for several reasons:

- studies often consider problems, representations and search algorithms that are relatively complex and so results represent the compositions of multiple effects (e.g., bloat or spurious attractors in genetic programming),
- there is not a single definition of neutrality and different studies have added neutrality to problems in radically different ways, and,
- the features of a problem's landscape change when neutrality is artificially added, but rarely an effort has been made to understand in exactly what ways.

Recently [14, 24], in an effort to shed some light on neutrality we started addressing these problems. In particular, we studied perhaps the simplest possible form of neutrality: a neutral network of constant fitness, identically distributed in the whole search space. For this form of neutrality, we analysed both problem-solving performance and population flows from and to the neutral network and the basins of attraction of the optima, as the fitness of the neutral network was varied.

In this paper, we will continue towards the same goals, but we will consider a much more practical form of neutrality, bit-wise neutrality.

3 Bitwise Neutrality

Bitwise neutrality is a form of neutrality induced by a genotype-phenotype map where each phenotypic bit is obtained by transforming a group of genotypic bits via some encoding function. In this paper we consider three different kinds of genotype-phenotype encodings to specify bitwise neutrality in the evolutionary process. For the three of them, each phenotypic bit is encoded using n genotypic bits.

These encodings are defined as follows:

1. The *majority* encoding works as follows: given n bits, the user defines a threshold (T) ($0 \leq T \leq n$) and if the number of ones that are in the n genotypic bits is greater or equal to T then the bit at the phenotype level is

set to 1, otherwise it is set to 0. Figure 1(a) illustrates this concept. Normally, to avoid biasing the system we will always use $T = n/2$ and n odd. This guarantees that 0s and 1s are treated identically.
2. The *parity* encoding works as follows: if the number of ones that are in n genotypic bits is an even number, then the bit at the phenotype level is set to 1, otherwise it is set to 0. Figure 1(b) illustrates this concept.
3. The *truth table* encoding works as follows: a truth table is generated and the output for each combination is produced at random. (Half of the outputs of the truth table are assigned with 0s and the other half are assigned with 1s. Then the outputs are shuffled to make them perfectly random). Then we consider the n genotypic bits as inputs, and we take as our phenotypic bit the corresponding truth table's output. Figure 1(c) illustrates this concept.

Neutrality is added to the non-redundant code by the proposed encodings. Because each bit is encoded using n bits, the same phenotype can be obtained from different genotypes and, so, neutrality is artificially added to the search space.

In the presence of the form of neutrality discussed above, the size of the search space is $2^{\ell n}$, where ℓ is the length of a phenotypic bit string and n is the number of bits required to encode each bit. With the types of encodings explained earlier, we have increased not only the size of the search space but also the size of the solution space. However, this does not mean that neutrality is always beneficial. We have also to bear in mind that the mutation rate at genotype level is different than the mutation rate at phenotype level. We will calculate these mutations rates and see their effects in Section 6.

Neutrality is often reported to help in multimodal landscapes, in that it can prevent a searcher from getting stuck in local optima. However, very little mathematical evidence to support this claim has been provided in the literature. So, in the next section we start our analysis by using a well-defined hardness measure, the fitness distance correlation, calculating it in such a way to make the dependency between problem difficulty and neutrality of the encoding explicit.

4 Fitness Distance Correlation

4.1 Definition

Jones [19, 20] proposed *fitness distance correlation* (fdc) to measure the difficulty of problem by studying the relationship between fitness and distance. The idea behind *fdc* was to consider fitness functions as heuristics functions and to interpret their results as indicators of the distance to the nearest global optimum of the search space and, so, *fdc* is an algebraic measure to express such a relationship.

The definition of *fdc* is quite simple: given a set $F = \{f_1, f_2, ..., f_n\}$ of fitness values of n individuals and the corresponding set $D = \{d_1, d_2, ..., d_n\}$ of distances to the nearest global optimum, we compute the correlation coefficient r, as:

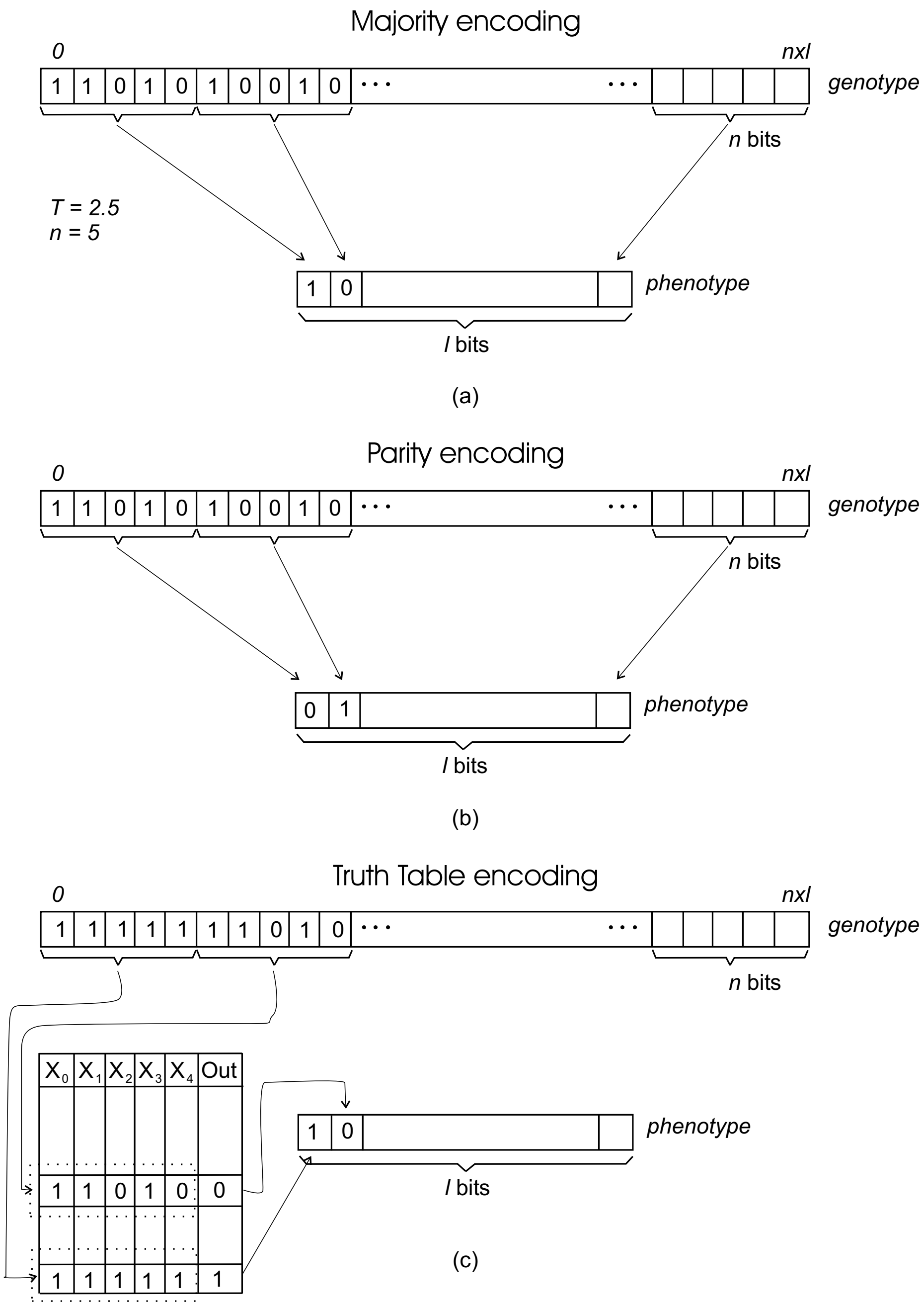

Fig. 1. Three different encodings used in our research: (a) Majority encoding, (b) Parity encoding and (c) Truth table encoding.

$$r = \frac{C_{FD}}{\sigma_F \sigma_D},$$

where:

$$C_{FD} = \frac{1}{n}\sum_{i=1}^{n}(f_i - \overline{f})(d_i - \overline{d})$$

is the covariance of F and D, and σ_F, σ_D, $\overline{f}$ and $\overline{d}$ are the standard deviations and means of F and D, respectively. The n individuals used to compute *fdc* can be chosen in different ways. For reasonably small search spaces or in theoretical calculations it is often possible to sample the whole search space. In this case *fdc* can be computed exactly. However, in most other cases, *fdc* is estimated by constructing the sets F and D via some form of random sampling.

According to [20] a problem can be classified in one of three classes, depending of the value of r: (1) *misleading* ($r \geq 0.15$), in which fitness tends to increase with the distance from the global optimum, (2) *difficult* ($-0.15 < r < 0.15$), for which there is no correlation between fitness and distance, and (3) *easy* ($r \leq -0.15$), in which fitness increases as the global optimum approaches.

There are some known weakness in the *fdc* as a measure of problem hardness [1, 26, 27]. However, it is fair to say that the method has been generally very successful [6, 20, 31, 34]. The distance used in the calculations is, for binary search spaces, the Hamming distance, H.

In this work we will use *fdc* to evaluate problem difficulty with and without neutrality. Here we only consider problems where the fitness function is a function of unitation, so, we can rewrite C_{FD} in a more explicit form.

4.2 Test Problems

We have used three problems to analyse neutrality. The first one is the OneMax problem. The problem is to maximise:

$$f(x) = \sum_i x_i,$$

where x is a binary string of length ℓ, i.e., $x \in \{0,1\}^\ell$. Naturally, this problem has only one global optimum in $11\cdots1$, and, the landscape is unimodal. Seen as a function of unitation the problem is represented by $f(u) = u$ or $f(x) = u(x)$ where $u(x)$ is a function that returns the unitation value of x.

For the second problem, we used a multimodal problem generator [8, 21, 23]. The idea is to create problem instances with a certain degree of multi-modality. In general, for a problem with P peaks, P bit strings of length ℓ are randomly generated. The generator works as follows. To evaluate an arbitrary individual x, we first locate the nearest peak in Hamming space

$$Peak_n(x) = \arg\min_i H(Peak_i, x)$$

In case there is a tie, the highest peak is chosen. The fitness of x is the number of bits the string has in common with that nearest peak, divided by ℓ and scaled by the height of the nearest peak:

$$f(x) = \frac{\ell - H(x, Peak_n(x))}{\ell} \times Height(Peak_n(x))$$

In this problem, the fitness value has a range from 0.0 and 1.0. The goal is to find the highest peak (i.e., to find a string with fitness 1.0). The difficulty of the problem depends on the number of peaks, the distribution of peaks and, finally, the distribution of peak heights. To carry out our experiments, we have tuned the parameters in such a way to make the problem much harder than the OneMax problem but easier than the trap function (see below). More details will be provided in Section 7.

The third and last problem is a Trap function, which is a deceptive function of unitation [15–17]. For this example, we have used the function:

$$f(x) = \begin{cases} \frac{a}{u_{min}}(u_{min} - u(x)) & \text{if } u(x) \leq u_{min}, \\ \frac{b}{\ell - u_{min}}(u(x) - u_{min}) & \text{otherwise} \end{cases}$$

where a is the deceptive optimum, b is the global optimum, and u_{min} is the slope-change location. Basically the idea is that there are two optima, a and b, and by varying the parameters ℓ and u_{min}, we can make the problem easier or harder.

4.3 *fdc* in the Absence of Neutrality

For all our test problems, given a search space of binary strings of length ℓ and being the unitation of the optimal string $u_{opt} = \ell$, if we sample the whole search space in order to compute C_{FD}, we have:

$$C_{FD} = \frac{1}{2^\ell} \sum_{u=0}^{\ell} \binom{\ell}{u} (f(u) - \overline{f})(\ell - u - \overline{d})$$

where:

$$\overline{f} = \frac{1}{2^\ell} \sum_{u=0}^{\ell} \binom{\ell}{u} f(u)$$

and

$$\overline{d} = \ell - \frac{1}{2^\ell} \sum_{u=0}^{\ell} \binom{\ell}{u} u = \frac{\ell}{2}.$$

Similar expressions can be obtained for σ_D and σ_F.

So, for example, for OneMax, where $f(u) = u$, we have $\bar{f} = \frac{\ell}{2}$ and

$$\begin{aligned} C_{FD} &= \frac{1}{2^\ell} \sum_{u=0}^{\ell} \binom{\ell}{u} \left(u - \frac{\ell}{2}\right)\left(\frac{\ell}{2} - u\right) \\ &= -\frac{\ell}{4} \end{aligned}$$

as one can easily see by noting that $\frac{1}{2^\ell}\binom{\ell}{u}$ is a binomial distribution, $\binom{\ell}{u}p^u(1-p)^{\ell-u}$, with success probability $p = 1/2$. Therefore, by definition of variance, $C_{FD} = -Var[u] = -\ell p(1-p) = -\frac{\ell}{4}$. By similar arguments one finds $\sigma_D^2 = \sigma_F^2 = \frac{1}{4}$, whereby $r = -1$, suggesting an easy problem. For Trap functions, instead, whenever $u_{min} \approx \ell$ one finds $r \approx 1$ [19] indicating hard problems.

5 *fdc* in the Presence of Bitwise Neutrality

As mentioned in Section 3, the form of neutrality we consider here is one where each phenotypic bit is encoded using n genotypic bits. In this situation, C_{FD} is given by:

$$C_{FD} = \frac{1}{2^{n\ell}} \sum_{x\in\{0,1\}^{n\ell}} (f_x(x) - \bar{f})(d(x) - \bar{d})$$

where $x = x_1 \cdots x_{n\ell}$ is a genotype and $f_x(x)$ is the genotypic fitness. Similar expressions can be obtained for σ_D and σ_F. Note that $f_x(x)$ can be written as

$$f_x(x) = f_y(g(x^{(1)}), g(x^{(2)}), \cdots, g(x^{(n)}))$$

where $x^{(k)} = x_{(k-1)n+1} \cdots x_{kn}$ is a substring of x, g is one of our encoding functions (e.g., Majority or Parity), and $f_y(y)$ is the phenotypic fitness ($y \in \{0,1\}^\ell$), which in this work we will assume to be a function of unitation.

We define $X_n = \{x \in \{0,1\}^n : g(x) = 1\}$ and $\bar{X}_n = \{x \in \{0,1\}^n : g(x) = 0\}$. We require that our encoding functions g respect one property: that on average they return as many 0s as 1s, i.e.,

$$\sum_{x\in\{0,1\}^n} g(x) = 2^{n-1}.$$

This property is respected by the encodings described in Section 3. So, $|X_n| = |\bar{X}_n| = 2^{n-1}$.

To illustrate the effects of the introduction of bitwise neutrality, in the following we will consider in detail the case of OneMax.

5.1 *fdc* for OneMax with Bitwise Neutrality

For the OneMax function we have

$$f_x(x) = \sum_i g(x^{(i)}).$$

To compute *fdc* we can make use of a result originally derived by Jones [19, Appendix D]: the concatenation of multiple copies of a problem does not change the *fdc* of the original problem, provided the fitness of the concatenated problem is obtained by summing the fitnesses of the sub-problems. This result is applicable because we can interpret g as the fitness functions of an n-bit problem which is concatenated ℓ times to form an $\ell \times n$ bit problem with fitness function $f_x(x)$. Therefore, we can compute the *fdc* for OneMax with different forms of bitwise neutrality by simply computing the *fdc* of the corresponding g functions. Since these functions take only binary values, this calculation is simpler than the original.

Let us start by considering the mean value of the function g, $\bar{g}$, for all encodings. By definition we have that $g(x) = 1$ for $x \in X_n$ and $g(x) = 0$ otherwise. So, irrespective of the encoding used we have

$$\bar{g} = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} g(x) = \frac{1}{2^n} \sum_{x \in X_n} 1 = \frac{1}{2^n} |X_n| = \frac{1}{2}$$

irrespective of the encoding function used.

We use this result in the computation of σ_F^2, obtaining

$$\begin{aligned} \sigma_F^2 &= \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (g(x) - \bar{g})^2 \\ &= \frac{1}{2^n} \left(\sum_{x \in X_n} \left(1 - \frac{1}{2}\right)^2 + \sum_{x \in \bar{X}_n} \left(0 - \frac{1}{2}\right)^2 \right) \\ &= \frac{1}{4}, \end{aligned}$$

which, again, is valid for all encodings.

Also, we have

$$\bar{d} = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} H(x, N(x))$$

where $N(x)$ is the global optimum nearest to x and H is the Hamming distance. Because all elements of X_n are global optima of g, and, so, $x = N(x)$ and $H(x, N(x)) = 0$ for $x \in X_n$, we have

$$\bar{d} = \frac{1}{2^n} \sum_{x \in \bar{X}_n} H(x, N(x)). \tag{1}$$

If we extend the definition of Hamming distance to sets by via the definition $H(x, S) = \min_{y \in S} H(x, y)$, we can rewrite Equation (1) as

$$\bar{d} = \frac{1}{2} E[H(x, X_n) | x \in \bar{X}_n], \tag{2}$$

where $E[H(x, X_n) | x \in \bar{X}_n]$ is the mean Hamming distance between the elements of $\bar{X}_n$ and the set X_n.

Similarly we have,

$$\begin{aligned}
\sigma_D^2 &= \frac{1}{2^n} \sum_{x\in\{0,1\}^n} (H(x,N(x)) - \bar{d}\,)^2 \\
&= \frac{1}{2^n}\left(\sum_{x\in X_n}\left(0-\bar{d}\,\right)^2 + \sum_{x\in\bar{X}_n}\left(H(x,N(x))-\bar{d}\,\right)^2\right) \\
&= \frac{1}{2}\left(\bar{d}^{\,2} + E\Big[\left(H(x,X_n)-\bar{d}\,\right)^2 \Big| x\in\bar{X}_n\Big]\right) \\
&= \frac{1}{2}\left(\bar{d}^{\,2} + E\Big[H(x,X_n)^2\Big|x\in\bar{X}_n\Big] - 2\bar{d}E\Big[H(x,X_n)\Big|x\in\bar{X}_n\Big] + \bar{d}^{\,2}\right) \\
&= \frac{1}{2}\left(\bar{d}^{\,2} + E\Big[H(x,X_n)^2\Big|x\in\bar{X}_n\Big] - 2\bar{d}\times(2\bar{d}) + \bar{d}^{\,2}\right) \\
&= \frac{1}{2}\left(E\Big[H(x,X_n)^2\Big|x\in\bar{X}_n\Big] - 2\bar{d}^{\,2}\right)
\end{aligned}$$

Finally, we have

$$\begin{aligned}
C_{FD} &= \frac{1}{2^n}\sum_{x\in\{0,1\}^n}(g(x)-\bar{g}\,)(H(x,N(x))-\bar{d}\,) \\
&= \frac{1}{2^n}\left(\sum_{x\in X_n}\left(1-\frac{1}{2}\right)\left(0-\bar{d}\,\right) + \sum_{x\in\bar{X}_n}\left(0-\frac{1}{2}\right)(H(x,N(x))-\bar{d}\,)\right) \\
&= \left(-\frac{1}{2}\right)\left(\frac{1}{2}\bar{d} + \frac{1}{2^n}\sum_{x\in\bar{X}_n}H(x,N(x)) - \frac{1}{2^n}\sum_{x\in\bar{X}_n}\bar{d}\right) \\
&= \left(-\frac{1}{2}\right)\left(\frac{1}{2^n}\sum_{x\in\bar{X}_n}H(x,N(x))\right) \\
&= \left(-\frac{1}{2}\right)\left(\frac{1}{2}E[H(x,N(x))|x\in\bar{X}_n]\right) \\
&= -\frac{\bar{d}}{2}
\end{aligned}$$

In the following subsections we apply these generic results to our three encoding functions: Parity, Truth Table and Majority.

5.2 *fdc* under Parity

Let us start with the Parity encoding. The bit strings in $\bar{X}_n$ have all odd parity. Therefore, they can be turned into even-parity global optima by a single bit flip. That is, their Hamming distance from a global optimum is always 1, whereby $E[H(x,X_n)|x\in\bar{X}_n] = 1$. So, from Equation (2) we obtain

$$\bar{d} = \frac{1}{2}.$$

From this, it follows that

$$C_{FD} = -\frac{1}{4}.$$

We also have that $E\Big[H(x,X_n)^2\Big|x \in \bar{X}_n\Big] = 1$. So,

$$\sigma_D^2 = \frac{1}{2}\left(1 - 2 \times \frac{1}{4}\right) = \frac{1}{4}.$$

Therefore, the fitness distance correlation for OneMax under the Parity encoding is

$$r = \frac{-\frac{1}{4}}{\sqrt{\frac{1}{4}}\sqrt{\frac{1}{4}}} = -1$$

That is, the *fdc* of OneMax is unaffected by the presence of bitwise neutrality under Parity encoding, irrespective of the number of bits (n) one uses.

5.3 *fdc* under Truth Table

Let us now consider the Truth Table encoding. In order to apply Equation (2) we need to compute $E[H(x,X_n)|x \in \bar{X}_n]$. To do this we will treat $H(x,X_n)$ as a stochastic variable, compute its probability distribution and then make use of the definition of expected value. Let us call $p(d)$ the probability that $H(x,X_n) = d$ for a randomly chosen $x \in \bar{X}_n$.

Let us choose uniformly at random an $x \in \bar{X}_n$ and then choose randomly one of the Hamming-1 neighbours, x', of x. Because the entries of the truth table are randomly assigned, the probability that $x' \in X_n$ is $\frac{1}{2}$. Note that $p(1)$ is the probability that at least one neighbour of x is a member of X_n. Since x has n neighbours and each neighbour's membership of X_n is a Bernoulli trial, we have that

$$p(1) = 1 - \left(\frac{1}{2}\right)^n.$$

So, as n grows, $p(1)$ rapidly approaches 1.

Let us now focus on $p(2)$. This can be seen as the probability of a joint event, i.e., none of the Hamming-1 neighbours of a randomly chosen $x \in \bar{X}_n$ is a member of X_n, but at least one of its Hamming-2 neighbours is. We treat these two events as independent.[1] We know that the probability of the first event is just $1 - p(1) = \left(\frac{1}{2}\right)^n$ and we compute the probability of the second as one minus the probability that none of the Hamming-2 neighbours of x is in X_n. Since there are $\binom{n}{2}$ such neighbours and the probability of each being in X_n is $\frac{1}{2}$, the

[1] This is an approximation, but its accuracy rapidly improves with n. So, our calculations are already very accurate for $n \geq 3$.

probability that none of the Hamming-2 neighbours of x is in X_n is $1 - \left(\frac{1}{2}\right)^{\binom{n}{2}}$. Putting everything together we then get

$$p(2) = \left(\frac{1}{2}\right)^n \left(1 - \left(\frac{1}{2}\right)^{\binom{n}{2}}\right)$$

Similarly, we get

$$p(3) = \left(\frac{1}{2}\right)^{n+\binom{n}{2}} \left(1 - \left(\frac{1}{2}\right)^{\binom{n}{3}}\right)$$

and, more generally,

$$p(d) = \left(\frac{1}{2}\right)^{\sum_{k=1}^{d-1} \binom{n}{k}} \left(1 - \left(\frac{1}{2}\right)^{\binom{n}{d}}\right).$$

We are now in a position to compute

$$E[H(x, X_n)|x \in \bar{X}_n] = \sum_{d=1}^{n} d \cdot p(d). \tag{3}$$

Note that $p(d)$ is a very rapidly decreasing function. For example, for $n = 4$ we have $p(1) = 0.93750$, $p(2) = 0.061523$, $p(3) = 0.00091553$ and $p(4) = 0.000030518$. Furthermore, as n increases more and more of the probability mass accumulates onto $p(1)$, effectively leading to only $p(1)$ and $p(2)$ having any relevance in the calculation in Equation (3). So, we can write

$$E[H(x, X_n)|x \in \bar{X}_n] \approx p(1) + 2p(2) = 1 + \left(\frac{1}{2}\right)^n - 2\left(\frac{1}{2}\right)^{n+\binom{n}{2}},$$

which makes it clear that for the Truth Table encoding $E[H(x, X_n)|x \in \bar{X}_n] \approx 1 + 2^{-n}$. For example, for $n = 3, 4, 5, 6$, and computing $E[H(x, X_n)|x \in \bar{X}_n]$ using Equation (3) we obtain the values 1.11719, 1.06342, 1.03128, 1.01563, respectively. So, under the Truth Table encoding

$$\bar{d} \approx \frac{1}{2} + 2^{-n-1}$$

From this, it follows that

$$C_{FD} = -\frac{1}{4} - 2^{-n-2}.$$

Using a similar approach we compute

$$\begin{aligned} E\Big[H(x, X_n)^2 \Big| x \in \bar{X}_n\Big] &= \sum_{d=1}^{n} d^2 \cdot p(d) \\ &\approx p(1) + 4p(2) \\ &= 1 + 3\left(\frac{1}{2}\right)^n - 4\left(\frac{1}{2}\right)^{n+\binom{n}{2}} \\ &\approx 1 + 3 \times 2^{-n} \end{aligned}$$

So,

$$\begin{aligned}\sigma_D^2 &\approx \frac{1}{2}\left(1 + 3 \times 2^{-n} - 2 \times \left(\frac{1}{2} + 2^{-n-1}\right)^2\right)\\ &\approx \frac{1}{2}\left(1 + 3 \times 2^{-n} - \frac{1}{2} - 2^{-n}\right)\\ &\approx \frac{1}{2}\left(\frac{1}{2} + 2 \times 2^{-n}\right)\\ &= \frac{1}{4} + 2^{-n}\end{aligned}$$

Therefore, the fitness distance correlation for OneMax under the Truth Table encoding is

$$r = -\frac{\frac{1}{4} + 2^{-n-2}}{\sqrt{\frac{1}{4} + 2^{-n}}\sqrt{\frac{1}{4}}} = -\frac{2^{(-n-1)} + \frac{1}{2}}{\sqrt{2^{-n} + \frac{1}{4}}} \approx -1 + 2^{-n}.$$

That is, for $n \geq 5$ or so, also Truth Table induces a form of neutrality which effectively leaves the *fdc*/problem difficulty unchanged. For relatively small values of n, however, this encoding makes the OneMax problem harder, albeit to a small degree. In Section 6 and 7 we will use $n \geq 5$, for which Truth Table effectively behaves like Parity.

5.4 *fdc* under Majority

Let us now consider the Majority encoding. Again, we start by computing $E[H(x, X_n)|x \in \bar{X}_n]$.

With a Majority encoding with $T = n/2$ and n odd, $\bar{X}_n$ is the class of all strings of length n which have 0, 1, ... $\lfloor T \rfloor$ bits set to 1. That is, we can naturally describe $\bar{X}_n$ by saying that it is contains all strings with unitation value $u < T$. Given a string in $\bar{X}_n$ having unitation u, we can compute how close this is to X_n just by looking at how many additional 1's would be needed to transform the string into a member of X_n. This number is simply $\lceil T - u \rceil$. Since for each unitation class u we have $\binom{n}{u}$ strings, we can then write

$$E[H(x, X_n)|x \in \bar{X}_n] = \frac{1}{2^{n-1}} \sum_{x \in \bar{X}_n} H(x, X_n) = \frac{1}{2^{n-1}} \sum_{u<T} \binom{n}{u} \times \lceil T - u \rceil.$$

This can be computed numerically. For $T = n/2$, n odd, and small values of n, $E[H(x, X_n)|x \in \bar{X}_n]$ grows approximately as $0.63 + 0.37\sqrt{n}$. So, we have

$$\bar{d} \approx 0.315 + 0.185\sqrt{n}$$

and

$$C_{FD} \approx -0.1575 - 0.0925\sqrt{n}.$$

Using a similar approach we compute

$$\begin{aligned} E\Big[H(x,X_n)^2\Big|x\in\bar{X}_n\Big] &= \frac{1}{2^{n-1}}\sum_{x\in\bar{X}_n} H(x,X_n)^2 \\ &= \frac{1}{2^{n-1}}\sum_{u<T}\binom{n}{u}\times\lceil T-u\rceil^2 \\ &\approx 0.725+0.334\times n \end{aligned}$$

for small values of n. So,

$$\begin{aligned} \sigma_D^2 &\approx \frac{1}{2}\Big(0.725+0.334\times n-2\left(0.315+0.185\sqrt{n}\right)^2\Big) \\ &\approx 0.133\,n-0.117\,\sqrt{n}+0.263 \end{aligned}$$

Therefore, the fitness distance correlation for OneMax under the Majority encoding is

$$r\approx-\frac{0.315+0.185\sqrt{n}}{\sqrt{0.133\,n-0.117\,\sqrt{n}+0.263}}.$$

So, in this case there is a much more marked effect of the encoding on the difficulty of a problem with the *fdc* progressively increasing (from the original value of -1) when n increases. For example, for $n = 3, 5, 7, 9, 11$ we obtain *fdc* values of approximately -0.9376, -0.8926, -0.8554, -0.8261 and -0.8028, respectively.

Naturally, theoretical *fdc* calculations could be performed also for the Multimodal problem generator and the Trap function in the presence of bitwise neutrality, although for these functions one could not use Jones' result [19, Appendix D]. We do not report these calculations. However, based on our results with OneMax and the results in [32], it is easy to understand that the Parity and Truth Table encodings have no or limited influence on the difficulty of the Trap and Multimodal functions. However, we should expect Majority to make these problems easier.

6 Phenotypic Mutation Rates

The analysis based on *fdc* indicates that the choice of encoding function used to introduce neutrality may be critical in determining whether a problem is made easier or harder by the introduction of neutrality in evolutionary search. However, fitness landscapes and *fdc* effectively neglect to model the fact that the precise distribution of mutants may have an important effect of search behaviour and performance. For example, *fdc* remains the same irrespective of the mutation probability p_{mut}.

So, to better evaluate benefits and drawbacks of neutrality we want to understand what effects different types of neutral encodings have on the way the search proceeds under mutation. In particular we are interested in understanding how genotypic mutations are related to phenotypic mutations, since only phenotypic changes can lead to fitness changes. To do so, we use the notion of *phenotypic mutation rate.*

When the parity encoding is used, the phenotypic mutation rate corresponding to a genotypic mutation rate p_{mut} is given by:

$$pmut_{phenotypic} = \sum_{i=1,3,5,\ldots} \binom{n}{i} p_{mut}^{i} (1 - p_{mut})^{n-i}$$

This is because only an odd number of genotypic bit-flips can produce a phenotypic change.

When the Truth Table encoding is used, the mutation rate at phenotype level is given by:

$$pmut_{phenotypic} = \frac{1 - (1 - p_{mut})^{n}}{2}$$

This is because there is the potential for a change in phenotypic value whenever we change the row from which we read out the output in the truth table. This happens if at least one genotypic mutation takes place (hence the factor $1 - (1 - p_{mut})^{n}$). However, not all row changes lead to a flipped phenotypic bit. Because the table is random, this happens only in 50% of the cases (hence the denominator, 2).

The calculation of the phenotypic mutation rates for Majority are more difficult. We can, however, obtain numerical estimates for these very easily. We do this by generating genotypic mutants of individuals using a particular genotypic mutation rate and recording how frequently the mutants are in a different majority class than the original parents.

In Table 1, we show the phenotypic mutation rates when mutation rates at genotype level are 0.01, 0.06 and 0.1 for Parity, Truth Table and Majority. In the case of Majority figures are estimates obtained by generating 10,000 mutants starting from a uniform random population. As we can see, there are conditions in which different encodings produce similar phenotypic mutation rates. This is the case, for instance, for the pairs of numbers in **boldface**, underlined, in *italics* and in sans serif. Note that the Parity and Truth Table (for the values of n used in the table) leave the fitness distance correlation of a problem unchanged, as discussed in the previous section. So, whenever also the phenotypic mutation rates match, we should expect to see similar performance under these two encodings. We will verify this in the next section.

7 Results and Analysis

For OneMax and the other two problems we have used chromosomes of length $\ell = 14$. For the multimodal landscape we have used $P = 400$ peaks. These were distributed in such a way to give the problem deceptive features. Specifically, the highest peak is at position $111 \cdots 111$ and the second highest peak is at

Table 1. Phenotypic mutation rates when mutation rates at genotype level are 0.01, 0.06 and 0.1.

Type of redundancy	$P_{mut} = 0.01$	$P_{mut} = 0.06$	$P_{mut} = 0.1$
Parity (n bits = 5)	0.0480	*0.2361*	0.3362
Parity (n bits = 6)	0.0571	0.2678	0.3689
Parity (n bits = 7)	0.0659	**0.2957**	0.3951
Parity (n bits = 8)	0.0746	0.3202	0.4161
Truth Table (n bits = 5)	0.0245	0.1331	<u>0.2048</u>
Truth Table (n bits = 6)	0.0293	0.1551	*0.2343*
Truth Table (n bits = 7)	0.0340	0.1758	0.2609
Truth Table (n bits = 8)	0.0386	<u>0.1952</u>	**0.2848**
Majority ($n = 5, T = 2.5$)	0.0168	0.0916	0.1530
Majority ($n = 7, T = 3.5$)	0.0204	0.1072	0.1725

Table 2. Parameters.

Parameter	*Value*
Length of the genome	14
Population Size	80
Generations	100
Mutation Rate (per bit)	0.01, 0.06, 0.1
Number of n bits encoded	5, 6, 7, 8
Independent Runs	1,000

position $000 \cdots 000$, the remaining peaks are randomly distributed. This last feature makes the problem easier than the trap function. For the trap function we used the following parameters: $u_{min} = 13$, $a = 39$, $b = 40$. Figure 2 depicts this trap function. For the three problems we have used a sample size 4,000 to calculate *fdc*.

The experiments were conducted using a GA with fitness proportionate selection and bit-flip mutation. Runs were stopped when the maximum number of generations was reached. The parameters used are given in Table 2.

Let's start by analysing *fdc* for the problems used in our experiments. Table 3 reports *fdc* for a representation without neutrality and for various forms of neutral encoding. As predicted in Section 4, for the three problems, the Parity and Truth Table encodings leave the *fdc* unchanged w.r.t. whatever value it had in the absence of neutrality.[2] On the contrary, as predicted, Majority moves slightly the *fdc* of a problem towards zero, thereby making easy problems harder and hard problems easier. The question now is: will actual search performance be similarly affected?

[2] This is not unexpected, since, as discussed in Section 2, the Parity encoding is a case of trivial neutrality (where the evolution of phenotypic bit strings can be modelled without referring to the corresponding genotypes). Also, the Truth Table encoding effectively becomes a case of trivial neutrality for sufficiently large n.

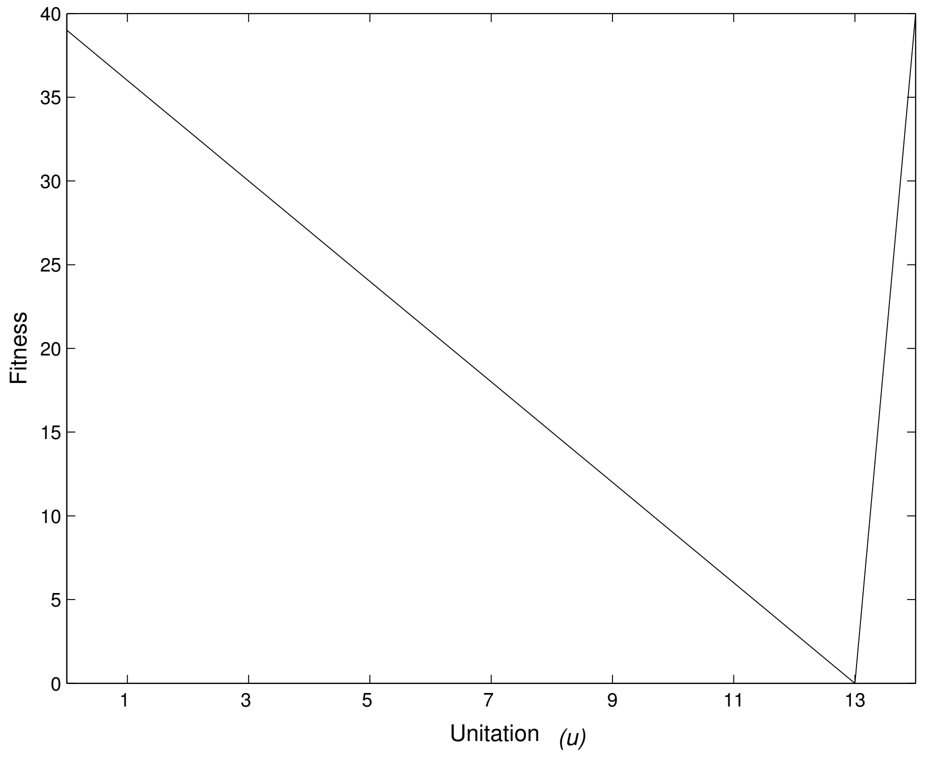

Fig. 2. The trap function used in our experiments ($u_{min} = 13$, $a = 39$, $b = 40$).

In Table 4, we show the average number of generations required to reach the optimum of OneMax and the percentage of successes in finding the optimum measured in 1,000 independent runs of a GA. Let us analyse these results.

When $p_{mut} = 0.01$ we can see a good match between the predictions of *fdc* and problem difficulty. In particular, the Parity and Truth Table encodings show almost exactly the same performance both in terms of percentage of runs where the problem was solved and average number of generations required to solve it. Also, we can see that, as predicted by our *fdc* analysis, the problem is easy and remains easy under all encodings, being solved in almost 100% of cases in all configurations. In addition, we can see that under Majority more generations are required to solve the problem than under Parity and Truth Table, which again confirms the predictions of the *fdc* analysis. There is, however, one element that is unexpected. In the absence of neutrality, runs take longer to find the optimum than with Parity and Truth Table. In fact, they take approximately as long as for Majority.

When $p_{mut} = 0.06$ the situation becomes less clear. Here Parity and Truth Table do not perform identically any more, with Truth Table still being able to solve the problem in almost all runs, while Parity does so only in between

Table 3. Fitness Distance Correlation estimated for the OneMax problem, the Multimodal Problem generator and the Trap function.

Type of redundancy	OneMax Problem	Multimodal Problem	Trap Function
No neutrality	-1	0.5114	0.9979
Parity ($n = 5$)	-1	0.5190	0.9925
Parity ($n = 6$)	-1	0.5190	0.9999
Parity ($n = 7$)	-1	0.5144	0.9999
Parity ($n = 8$)	-1	0.5086	0.9999
Truth Table ($n = 5$)	-0.9999	0.5102	0.9999
Truth Table ($n = 6$)	-1	0.5374	0.9925
Truth Table ($n = 7$)	-1	0.5264	0.9999
Truth Table ($n = 8$)	-0.9999	0.5233	0.9925
Majority ($n = 5, T = 2.5$)	-0.8488	0.4444	0.8434
Majority ($n = 7, T = 3.5$)	-0.8308	0.4471	0.8308

70 and 90% of the cases. This was not predicted by the *fdc* analysis. What is particularly surprising here is that in all cases Parity and Truth Table take longer to solve the problem than Majority and the no-neutrality case. So, Parity and Truth Table effectively make the problem harder, while the other two encodings are still performing approximately the same and their performance seems to be unaffected by the increase in mutation rate. *fdc* analysis also did not predict that performance would vary with n when using the Parity encoding.

These rather confusing trends continue also at the highest genotypic mutation rate, $p_{mut} = 0.1$. Now also the performance with Truth Table varies with n. Furthermore, in the no-neutrality case the problem is now solved in fewer generations than with the Majority encoding.

In summary, it is clear that while *fdc* captures some of the characteristics of a problem in relation to its difficulty for a GA, it does not capture all.

To explain these results one really needs to look at our second descriptor: the phenotypic mutation rates. As one can see in Table 1, when $p_{mut} = 0.01$ the encodings considered induce phenotypic mutation rates in the range 1.5-7.5%. At these mutation rates the GA solves the problem almost equally easily as it does without neutrality.

The more the phenotypic mutation rate is increased, the more the search will be expected to become undirected and random, leading to a worsening of performance. Indeed, when the genotypic mutation rate is increased to 0.06, the Truth Table encoding provides a phenotypic mutation rate which is significantly smaller than for Parity (see Table 1, second column). As expected, in these conditions the performance with Parity is worse than with Truth Table (see Table 4). The phenotypic mutation rates for Majority are even smaller than for Truth Table. So, it is not surprising to see that the GA performs better with Majority than with all other encodings.

When $p_{mut} = 0.1$, the phenotypic mutation rates for all encodings are further increased, leading to an even more undirected search. Note how, in these

Table 4. Performance of a mutation-based GA on the OneMax problem. Pairs of numbers in **boldface**, underline, *italics* or sans serif represent situations with almost identical phenotypic mutation rates.

	$p_{mut} = 0.01$		$p_{mut} = 0.06$		$p_{mut} = 0.1$	
	Avr. Gen	% Suc.	Avr. Gen	% Suc.	Avr. Gen	% Suc.
No neutrality	21.35	100%	14.39	100%	16.58	100%
Parity ($n = 5$)	14.55	100%	36.06	*90.1%*	44.02	62.7%
Parity ($n = 6$)	14.46	100%	38.38	82.6%	45.14	54.4%
Parity ($n = 7$)	14.49	100%	40.09	**73.3%**	42.12	49.7%
Parity ($n = 8$)	15.06	100%	43.26	68.2%	44.56	47.6%
Truth Table ($n = 5$)	16.63	99.9%	20.02	99.5%	29.21	95.0%
Truth Table ($n = 6$)	16.89	100%	22.87	99.4%	33.14	*90.5%*
Truth Table ($n = 7$)	15.89	100%	24.41	97.5%	35.49	84.5%
Truth Table ($n = 8$)	15.01	100%	28.16	97.4%	38.89	**78.8%**
Majority ($n = 5, T = 2.5$)	23.39	99.8%	17.26	99.7%	22.08	99.3%
Majority ($n = 7, T = 3.5$)	23.51	99.8%	17.93	100%	22.50	98.6%

conditions, the phenotypic mutation rates for Truth Table are similar to those observer at $p_{mut} = 0.06$ for Parity, and how performance is similar for these two cases (see Table 4). Similar phenotypic mutation rates and similar performance can also be observed for Majority (at $p_{mut} = 0.1$) and Truth Table (at $p_{mut} = 0.06$). At a mutation rate of 0.1, Parity presents high phenotypic mutation rates, reaching 41.6% in the case $n = 8$. In these conditions the search is almost random and so performance is poor.

Increasing further the genotypic mutation rate will lead the Parity and Truth Table encodings near a phenotypic mutation rate of 50%. There the search is effectively a random search. We do 8,000 trials (80 individuals for 100 generations) in each run, which represent 48.82% of the search space size, 2^ℓ, since $\ell = 14$. However, because of resampling we should only expect to find the optimum with probability 38.3%. (This can be computed using the theory for the coupon collector problem, see [11].) This is the limit performance for high mutation rates.

Let us now consider our second problem: the multimodal problem. For this problem, we tuned the parameters in such a way to make the problem hard, but still easier than the trap problem. Again, at the lowest mutation rate, the predictions of *fdc* are roughly correct: the problem is hard (*fdc* > 0) and remains hard irrespective of the encoding used and Parity and Truth Table lead to the same level of difficulty. Again, however, at the higher mutation rates the situation becomes rather more confusing, with Parity showing improved performance over the other encodings and a dependency of performance on n. Effectively, we can observe the opposite effects as in the OneMax problem. However, the confusion again disappears if we look at the mutation rates corresponding to each encoding.

Finally, let us consider the Trap problem. For this problem, the bigger the value of the slope-change location u_{min}, the harder the problem. In our experiments we chose $\ell = 14$ and $u_{min} = 13$ and, so, the problem is very hard. The behaviour of the evolutionary search in this problem is a mirror image of

Table 5. Performance of a GA on the Multimodal function. Pairs of numbers in **boldface**, underline, *italics* or sans serif represent situations with almost identical phenotypic mutation rates.

	$p_{mut} = 0.01$		$p_{mut} = 0.06$		$p_{mut} = 0.1$	
	Avr. Gen	% Suc.	Avr. Gen	% Suc.	Avr. Gen	% Suc.
No neutrality	8.56	3.2%	5.22	2.7%	11.54	1.9%
Parity ($n = 5$)	5.61	3.4%	41.2	*5.8%*	44.07	14.2%
Parity ($n = 6$)	4.76	3.4%	45.27	7.2%	50.41	19.4%
Parity ($n = 7$)	2.80	2.1%	44.41	**9.9%**	46.31	24.6%
Parity ($n = 8$)	4.85	2.1%	42.14	12.7%	46.94	23.2%
Truth Table ($n = 5$)	6.41	3.6%	15.86	2.5%	34.11	3.5%
Truth Table ($n = 6$)	8.18	2.5%	20.27	2.2%	34.32	*4.8%*
Truth Table ($n = 7$)	6.59	2.6%	24.07	3.1%	44.44	5.6%
Truth Table ($n = 8$)	4.95	3.6%	19.10	3.2%	33.03	**7.9%**
Majority ($n = 5, T = 2.5$)	11.41	2.0%	23.6	1.4%	15.62	1.9%
Majority ($n = 7, T = 3.5$)	9.76	2.3%	9.44	2.2%	25.42	2.4%

that observed on the OneMax problem (see Table 6). Again, we can see how *fdc* makes reasonably good predictions of relative difficulty under different encodings when $p_{mut} = 0.01$, but that the picture becomes less and less clear as p_{mut} increases. However, again, we can explain performance differences easily by looking at phenotypic mutation rates. In this case, because the problem is deceptive, the more random the search is, the more likely the global optimum is found. So, performance improves as the phenotypic mutation rate increases.

The importance of considering the phenotypic mutation rates instead of the classical genotypic mutation rates is shown in Figures 3 and 4. These figures simply plot the success probabilities reported in Tables 4, 5 and 6 against the corresponding genotypic and phenotypic mutation rates, totally ignoring distinctions between encodings. It is clear how the data strongly correlate with the phenotypic mutation rates, while they correlate much more weakly with the genotypic mutation rates. Note, for example, how the fairly ordinary genotypic mutation rate of 0.1 leads the GA to perform very close to the random search limit (38.3%). We believe this is one of the reasons why so much confusion is present in the EC literature on neutrality.

From these results, it is apparent that *fdc* roughly provides an indication of difficulty, but also that in order to obtain more accurate information one needs to consider how the chosen representation translates genotypic mutation rates into phenotypic mutation rates. With this information in hand, one should then expect to see that for problems with negative *fdc*, performance degrades as the phenotypic mutation rate increases, while the opposite happens for problems with positive *fdc*.

Table 6. Performance of a GA on the Trap function. Pairs of numbers in **boldface**, underline, *italics* or sans serif represent situations with almost identical phenotypic mutation rates.

	$p_{mut} = 0.01$		$p_{mut} = 0.06$		$p_{mut} = 0.1$	
	Avr. Gen	% Suc.	Avr. Gen	% Suc.	Avr. Gen	% Suc.
No neutrality	0.6	0.3%	7.2	0.7%	4.55	0.7%
Parity ($n = 5$)	1	0.5%	47.77	*10.4%*	44.85	22.0%
Parity ($n = 6$)	1	0.8%	45.96	15.6%	44.73	23.8%
Parity ($n = 7$)	1	0.6%	48.62	**15.4%**	46.82	32.0%
Parity ($n = 8$)	13.57	0.7%	46.27	20.2%	46.69	31.5%
Truth Table ($n = 5$)	1	0.7%	13.05	1.4%	41.49	6.3%
Truth Table ($n = 6$)	1.25	0.6%	35.16	2.1%	47.19	*7.8%*
Truth Table ($n = 7$)	1	0.1%	32.36	3.5%	47.32	10.9%
Truth Table ($n = 8$)	1	0.9%	34.44	4.8%	58.54	**13.0%**
Majority ($n = 5, T = 2.5$)	1	1.1%	4.4	1.2%	19.91	2.3%
Majority ($n = 7, T = 3.5$)	1	0.5%	1.16	0.6%	28.15	1.9%

8 Conclusions

In the literature there is contradicting evidence as to whether or not neutrality aids evolutionary search. We believe that, with notable exceptions (e.g., [32]), the confusion often derives from the fact that different researchers use radically different types of neutral encodings and neglect to consider the effect of important parameters such as the rate of application of genetic operators. As we have shown in this paper, small changes in the representation used and search parameters can turn a neutral encoding from being beneficial to being strongly disadvantageous and *vice versa*.

In this paper we considered a form of neutrality induced by a genotype-phenotype map where each phenotypic bit is obtained by transforming a group of genotypic bits via an encoding function. By using explicit calculations for fitness distance correlation, we showed under what conditions a neutral encoding has the potential to induce big changes in problem hardness. We also studied how phenotypic mutation rates change as a function of the genotypic mutation rate for different encodings. We then performed extensive empirical experimentation. We showed that the performance of a GA can change radically with different types of neutrality and mutation rates. However, phenotypic mutation rates and *fdc* allowed us to formulate simple explanations for why this happens.

Acknowledgements

The first author thanks the Mexican Consejo Nacional de Ciencia y Tecnología (CONACyT) for support to pursue graduate studies at University of Essex. The authors would like to thank the anonymous reviewers and Chris Stephens for their valuable comments. Marc Toussaint is particularly thanked for spotting a

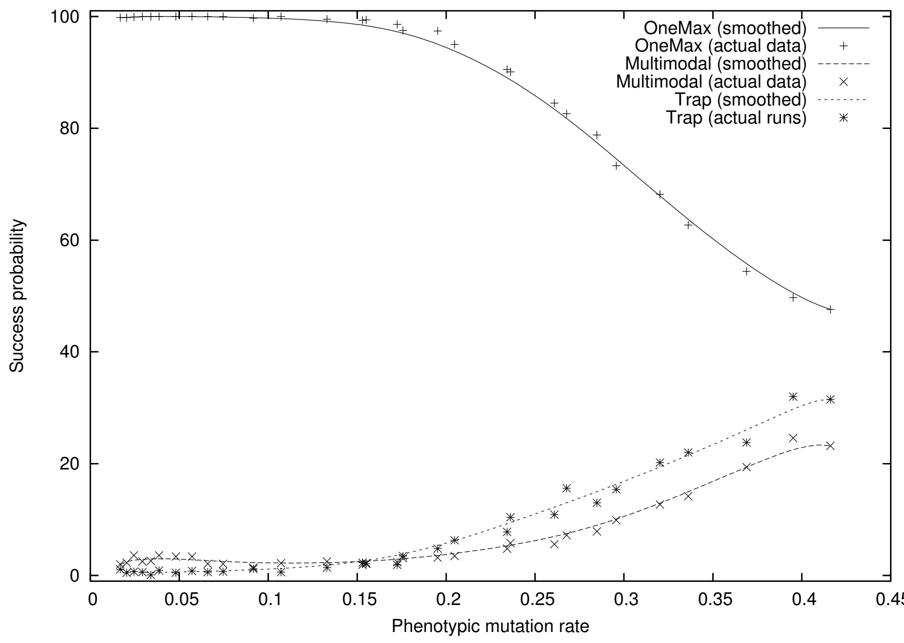

Fig. 3. Success probability vs. phenotypic mutation rates for the OneMax, Multimodal and Trap fitness functions. The points are obtained by combining the entries in Table 1 with those in Tables 4, 5 and 6.

serious problem in our origianal analysis, for his helpful suggestions and for his costant support throughout the revision of this manuscript.

References

1. L. Altenberg. Fitness distance correlation analysis: An instructive counterexample. In *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 57–64, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.
2. W. Banzhaf and A. Leier. Evolution on Neutral Networks in Genetic Programming. In T. Yu, R. L. Riolo, and B. Worzel, editors, *Genetic Programming Theory and Practice III*, volume 9 of *Genetic Programming*, chapter 14, pages 207–221. Springer, Ann Arbor, 12-14 May 2005.
3. L. Barnett. Ruggedness and neutrality – the NKp family of fitness landscapes. In C. Adami, R. K. Belew, H. Kitano, and C. Taylor, editors, *Artificial Life VI: Proceedings of the Sixth International Conference on Artificial Life*, pages 18–27, Cambridge, MA, 1998. MIT Press.
4. H. Beyer. *The Theory of Evolution Strategies*. Springer Verlag, 2001.
5. R. Chow. Effects of Phenotypic Feedback and the Coupling of Genotypic and Phenotypic Spaces in Genetic Searchers. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, pages 242–249, Portland, Oregon, 20-23 June 2004. IEEE Press.

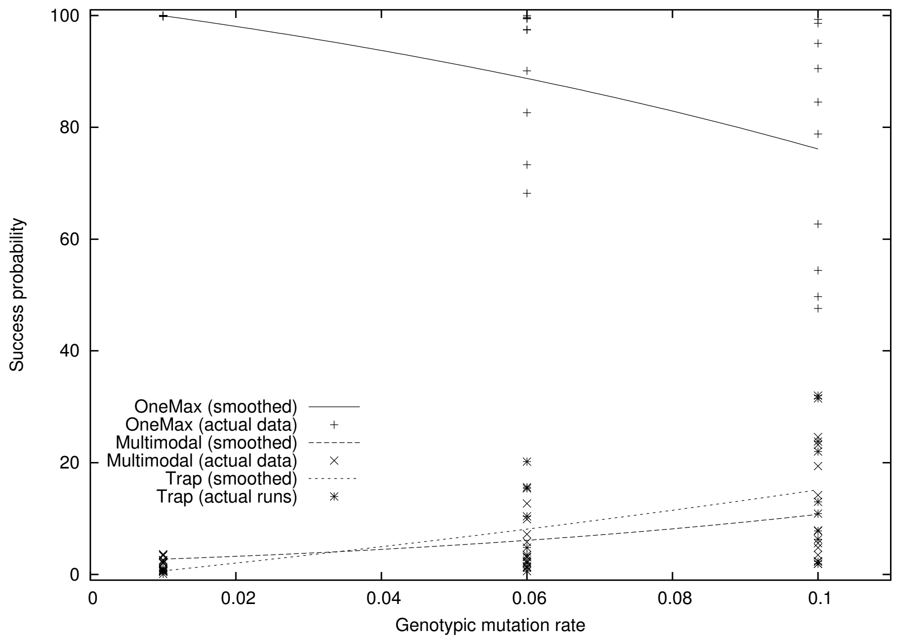

Fig. 4. Success probability vs. genotypic mutation rates for the OneMax, Multimodal and Trap fitness functions. The points are obtained from Tables 4, 5 and 6.

6. M. Clergue, P. Collard, M. Tomassini, and L. Vanneschi. Fitness distance correlation and problem difficulty for genetic programming. In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 724–732, New York, 9-13 July 2002. Morgan Kaufmann Publishers.
7. M. Collins. Finding needles in haystacks is harder with neutrality. In H.-G. B. *et al.*, editor, *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*, volume 2, pages 1613–1618, Washington DC, USA, 25-29 June 2005. ACM Press.
8. K. De Jong, M. A. Potter, and W. M. Spears. Using Problem Generators to Explore the Effects of Epistasis. In T. Back, editor, *Proceedings ICGA 1997: International Conference on Genetic Algorithms*, pages 338–345, San Francisco, USA, 1997. Morgan Kaufmann.
9. M. Ebner, P. Langguth, J. Albert, M. Schakleton, and R. Shipman. On Neutral Networks and Evolvability. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 1–8, Seoul, Korea, 27-30 May 2001. IEEE Press.
10. A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Trans. Evolutionary Computation*, 3(2):124–141, 1999.
11. W. Feller. *An Introduction to Probability Theory and Its Applications.* Wiley, 1968.
12. C. Fonseca and M. Correia. Developing Redudant Binary Representations for Genetic Search. In *Proceedings of the 2005 IEEE Congress on Evolutionary Com-*

putation (CEC-2005), pages 372–379, Edinburgh, 2-4 Sept. 2005. IEEE.

13. W. Fontana and P. Schuster. Continuity in evolution: On the nature of transitions. *Science*, 280:1431–1433, 1998.
14. E. Galván-López and R. Poli. An Empirical Investigation of How and Why Neutrality Affects Evolutionary Search. In *GECCO 2006: Proceedings of the 2006 conference on Genetic and evolutionary computation*, pages 1149–1156, Seattle, WA, USA, 8-12 July 2006. ACM Press.
15. D. E. Goldberg. Construction of high-order deceptive functions using low-order walsh coefficients. *Ann. Math. Artif. Intell.*, 5(1):35–47, 1992.
16. D. E. Goldberg, K. Deb, and J. Horn. Massive multimodality, deception, and genetic algorithms. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, Amsterdam, 1992. Elsevier Science Publishers, B. V.
17. K. D. H. Kargupta and D. Goldberg. Ordering genetic algorithms and deception. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, Amsterdam, 1992. Elsevier Science Publishers, B. V.
18. M. A. Huynen. Exploring phenotype space through neutral evolution. *Molecular Evolution*, 43:165–169, 1996.
19. T. Jones. *Evolutionary Algorithms, Fitness Landscapes and Search.* PhD thesis, University of New Mexico, Albuquerque, 1995.
20. T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 184–192, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.
21. J. Kennedy and W. M. Spears. Matching Algorithms to Problems: An Experimental Test of the Particle Swarm and Some Genetic Algorithms to the Multimodal Problem Generator. In *Proceedings IEEE International Conference on Evolutionary Computation*, pages 78–83, Piscataway, NJ, 1998. IEEE Press.
22. M. Kimura. Evolutionary rate at the molecular level. In *Nature*, volume 217, pages 624–626, 1968.
23. G. Lobo and C. F. Lima. On the Utility of the Multimodal Problem Generator for Assessing the Performance of Evolutionary algorithms. In *GECCO 2006: Proceedings of the 2006 conference on Genetic and evolutionary computation*, pages 1233–1240, Seattle, WA, USA, 8-12 July 2006. ACM Press.
24. E. G. López and R. Poli. Some steps towards understanding how neutrality affects evolutionary search. In T. P. Runarsson, H.-G. Beyer, E. K. Burke, J. J. M. Guervós, L. D. Whitley, and X. Yao, editors, *PPSN*, volume 4193 of *Lecture Notes in Computer Science*, pages 778–787. Springer, 2006.
25. J. F. Miller and P. Thomson. Cartesian genetic programming. In R. P. *et al.*, editor, *Genetic Programming, Proceedings of EuroGP'2000*, volume 1802 of *LNCS*, pages 121–132, Edinburgh, 15-16 Apr. 2000. Springer-Verlag.
26. B. Naudts and L. Kallel. A comparison of predictive measures of problem difficulty in evolutionary algorithms. *IEEE Transactions Evolutionary Computation*, 4(1):1–15, 2000.
27. R. J. Quick, V. J. Rayward-Smith, and G. D. Smith. Fitness distance correlation and ridge functions. In *PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, pages 77–86, London, UK, 1998. Springer-Verlag.
28. F. Rothlauf and D. Goldberg. Redundant representations in evolutionary algorithms. *Evolutionary Computation*, 11(4):381–415, 2003.

29. T. Smith, P. Husbands, and M. O'Shea. Neutral networks and evolvability with complex genotype-phenotype mapping. *Lecture Notes in Computer Science*, 2159:272–282, 2001.
30. T. Smith, P. Husbands, and M. O'Shea. Neutral networks in an evolutionary robotics search space. In *Congress on Evolutionary Computation: CEC 2001*, pages 136–145. IEEE Press, 2001.
31. M. Tomassini, L. Vanneschi, P. Collard, and M. Clergue. A study of fitness distance correlation as a difficulty measure in genetic programming. *Evolutionary Computation*, 13(2):213–239, Summer 2005.
32. M. Toussaint. On the evolution of phenotypic exploration distributions. In C. Cotta, K. De Jong, R. Poli, and J. Rowe, editors, *Foundations of Genetic Algorithms 7 (FOGA 2003)*, pages 169–182. Morgan Kaufmann, 2003.
33. M. Toussaint and C. Igel. Neutrality: A necessity for self-adaptation. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002)*, pages 1354–1359, 2002.
34. L. Vanneschi, M. Tomassini, P. Collard, and M. Clergue. Fitness distance correlation in structural mutation genetic programming. In C. Ryan, T. Soule, M. Keijzer, E. P. K. Tsang, R. Poli, and E. Costa, editors, *EuroGP*, volume 2610 of *Lecture Notes in Computer Science*, pages 455–464. Springer, 2003.
35. V. K. Vassilev and J. F. Miller. The advantages of landscape neutrality in digital circuit evolution. In *Proceedings of the Third International Conference on Evolvable Systems*, pages 252–263. Springer-Verlag, 2000.
36. K. Weicker and N. Weicker. Burden and Benefits of Redundancy. In W. Martin and W. Spears, editors, *Foundations of Genetic Algorithms 6*, pages 313–333, San Francisco, 2000. Morgan Kaufmann.
37. T. Yu and J. Miller. Neutrality and the evolvability of boolean function landscape. In *Fourth European Conference on Genetic Programming*, pages 204–211. Springer-Verlag, 2001.
38. T. Yu and J. F. Miller. Needles in haystacks are not hard to find with neutrality. In J. A. F. *et al.*, editor, *Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002*, volume 2278 of *LNCS*, pages 13–25, Kinsale, Ireland, 3-5 Apr. 2002. Springer-Verlag.
39. T. Yu and J. F. Miller. The role of neutral and adaptive mutation in an evolutionary search on the onemax problem. In E. Cantú-Paz, editor, *Late Breaking Papers at the Genetic and Evolutionary Computation Conference (GECCO-2002)*, pages 512–519, New York, NY, July 2002. AAAI.