The Effects of Constant and Bit-Wise Neutrality on Problem Hardness, Fitness Distance Correlation and Phenotypic Mutation Rates

Riccardo Poli and Edgar Galván-López

Abstract

Kimura's neutral theory of evolution has inspired researchers from the evolutionary computation community to incorporate neutrality into Evolutionary Algorithms (EAs) in the hope that it can aid evolution. The effects of neutrality on evolutionary search have been considered in a number of studies, the results of which, however, have been highly contradictory. In this paper, we analyse the reasons for this and we make an effort to shed some light on neutrality by addressing them. We consider two very simple forms of neutrality: constant neutrality — a neutral network of constant fitness, identically distributed in the whole search space — and bit-wise neutrality, where each phenotypic bit is obtained by transforming a group of genotypic bits via an encoding function. We study these forms of neutrality both theoretically and empirically (both for standard benchmark functions and a class of random MAX-SAT problems) to see how and why they influence the behaviour and performance of a mutation-based EA. In particular, we analyse how the fitness distance correlation of landscapes changes under the effect of different neutral encodings and how phenotypic mutation rates vary as a function of genotypic mutation rates. Both help explain why the behaviour of a mutation-based EA may change so radically as problem, form of neutrality and mutation rate are varied.

Index Terms

Neutrality, Phenotypic Mutation Rates, Problem Hardness, Fitness Distance Correlation, MAX-SAT, Genotype-Phenotype Mappings.

I. Introduction

Despite the proven effectiveness of Evolutionary Algorithms (EAs), they have also limitations. Researchers have attempted to make EAs more powerful by using a variety of approaches. Following the ideas of Kimura's neutral

Riccardo Poli is with the School of Computer Science and Electronic Engineering, University of Essex, UK e-mail: rpoli@essex.ac.uk

E. Galván-López is a visiting researcher with the School of Computer Science and Electronic Engineering, University of Essex, UK e-mail:edgar.galvan@gmail.com

theory of molecular evolution [1], [2], one strategy to achieve this has been the use of *neutrality* in EAs.

Kimura's theory states that the majority of evolutionary changes at molecular level are the result of random fixation of selectively neutral mutations. A mutation from one gene to another is neutral if it does not affect the phenotype. Thus, most mutations that take place in natural evolution are neither advantageous nor disadvantageous for the survival of individuals. It is then reasonable to extrapolate that, if this is how evolution has managed to produce the amazing complexity and adaptations seen in nature, then surely neutrality should aid also EAs. However, despite the numerous publications in this field, quite often there is confusion with regard to what neutrality is and, certainly, there are no general conclusions on the effects of neutrality.

Many contradictory results on neutrality in EAs have been reported. For instance, in [3], Yu and Miller performed runs using the well-known Cartesian Genetic Programming (CGP) system [4], [5] and even-*n*-parity Boolean problems with different degrees of difficulty ($n = \{5, 8, 10, 12\}$). They compared performance with and without neutrality and reported that their system performed better when neutrality was present. However, a few years later, Collins claimed the opposite [6], explaining that Yu and Miller's chosen problem class (the parity problems) was unusual and unsuitable for analysing neutrality using CGP. This is because both the landscape and the form of representation used have a high degree of neutrality and these make the drawing of general conclusions on the effects of neutrality difficult. These are just two authoritative examples[1] of publications available in the specialised literature which show controversial results on neutrality.

We believe that the confusion regarding neutrality is due to several reasons. These include the following:

- there is a lack of mathematical frameworks that explain how and why neutrality affects evolution;
- many studies have based their conclusions on performance statistics (i.e., on whether or not a system with neutrality could solve a particular problem faster or better than a system without neutrality), rather than a more in-depth analysis based on problem hardness measures and search characteristics;
- studies have often considered problems, representations and search algorithms that are relatively complex; as a consequence, results represent the compositions of multiple effects (e.g., bloat or spurious attractors in genetic programming [7], [8]);
- there is not a single definition of neutrality, and different studies have added neutrality to systems in radically different ways;
- very often studies focused their attention on particular 'properties' of neutrality without properly defining them; and
- the features of a problem's landscape and the behaviour of the search operators change when neutrality is artificially added, but rarely has an effort been made to understand in exactly what ways.

The main goal of this paper is to start shedding some light on neutrality by addressing the sources of confusion mentioned previously. The core elements in this work are:

[1]Both [3] and [6] were nominated as best papers in their conference tracks.

- Two very simple types of neutrality for EAs will be defined, starting from the simplest possible form, *constant neutrality*, and then moving to the more common *bitwise neutrality*.
- *Fitness distance correlation* will be used to analytically quantify the hardness of problems with certain characteristics (i.e., landscape features) in the presence and in the absence of neutrality.
- We will define and study *phenotypic mutation rates* in relation to corresponding genotypic mutation rates for different encodings.
- We will combine and corroborate the theory with empirical results using both classical benchmark problems and a class of MAX-3-SAT problems.

This paper is organised as follows. In the following section, previous work on neutrality will be presented. In Section III we will provide basic notions on fitness distance correlation. In Section IV, we describe the forms of neutrality – *constant* and *bitwise neutrality* – studied in this paper and we also introduce our test problems. In Section V, we study theoretically the effects of constant and bitwise neutrality on the difficulty of our test problems. In Section VI, we make the mathematical relationship between genotypic mutation rates and phenotypic mutation rates explicit. Section VII reports empirical results that confirm our theoretical analyses. In Section VIII, we discuss these results from both the point of view of understanding neutrality and the practical consequences of the work in relation to the solution of real-world problems. Finally, Section IX draws some conclusions.[2]

II. Previous Work on Neutrality

As mentioned previously, there is not a single definition of neutrality. Neutrality is implicitly defined as the presence of *neutral networks* in the search space. A neutral network is sometimes defined as a set of points in the search space with identical fitness. More often neutral networks are, instead, defined as sets of points in the search space having identical fitness *and* such that starting from any point in a set one can reach any other point in the set through one or more mutations (without ever leaving the set).

To clarify this, let us focus on a binary EA for simplicity. A solution s' is considered to be a nearest neighbour of a solution s, if s' is one unit of Hamming distance away from s. The set of neighbours of s is denoted by $V(s)$. If f is the fitness function, an $s' \in V(s)$ such that $f(s) = f(s')$ is a *neutral neighbour* of s. A neutral network is a set of solutions of identical fitness which is closed under the application of V.

Nimwegen *et al.* [13] suggested that neutrality appears automatically throughout the evolutionary process. They focused their attention on how the population moves through neutral networks and suggested that the population does not drift purely randomly through them. Instead, the majority of individuals tend to migrate and stay in highly connected parts of the network (i.e., areas where points have a high number of neutral neighbours). This results in phenotypes that are relatively robust against mutations.

In the same vein, Wagner [14] argued that the presence of neutrality in a system makes it more robust against mutations. Moreover, Wagner suggested that neutrality should be viewed as an element that promotes evolvability

[2]Preliminary versions of the work presented in this paper have appeared in workshops and conferences papers [9]–[11], while other results can be found in Edgar Galván-López's PhD thesis [12].

and can help to discover new phenotypes. Obviously, as Wagner pointed out, neutrality in itself cannot offer any benefit: by definition, a neutral mutation at genotype level does not change the phenotype. However, neutrality can still be of help in that it allows evolutionary search to visit previously unexplored areas.

Reidys and co-workers [15] studied the relationship between RNA sequence and secondary structure, which is seen as mapping from sequence space into shape space [16]. The mapping has a high degree of redundancy (i.e., there are many more sequences than structures). In that work, the authors suggested that identical phenotypic structures form a neutral network if the *fraction* of neutral nearest neighbours exceeds a certain threshold. This is in sharp contrast with the definition used in most other work, where solutions are considered to form a neutral network if they are one unit of Hamming distance away from each other.

Toussaint and Igel [17] pointed out that standard approaches to self-adaptation in EAs (e.g., see [18]) are an explicit example of the benefit of neutrality. In these approaches the genome is augmented with strategy parameters, which typically describe the mutation distribution (e.g., the mutation rate). These are neutral parts of the genome which are co-adapted during evolution so as to induce better search distributions. The point of view developed in [17] is that the core aspect of neutrality is that different genomes in a neutral set provide a variety of different mutation distributions. Evolution may choose from these in a self-adaptive way. Interestingly, theoretical work on the evolution of strategy parameters (e.g., see [19]) can be re-interpreted as dealing with the evolution of neutral traits.

This line of thought was further formalised in [20]. Given a fixed genotype-phenotype mapping one can investigate the variety 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 showed that trivial neutrality is a necessary and sufficient condition for compatibility [21] with phenotypic projection of a mutation-selection EA. In other words, whether one or another representative of a neutral set is present in a population does not influence the evolution of phenotypes. Intuitively, this means that, in the presence of trivial neutrality, neutral traits have no effect on phenotypic evolution. In the case of non-trivial neutrality, different genotypes in a neutral set induce different phenotypic distributions. This implies a selection between equivalent genotypes similar to the selection of strategy parameters in self-adaptive EAs. Toussaint interpreted this as the underlying mechanism of the evolution of genetic representations.

In [22], [23], Shackleton *et al.* artificially added neutrality to evolutionary search with the use of five different genotype-phenotype mappings. The *static random mapping* consisted in defining a genotype of length 30 which was mapped to a phenotype of 16 bits. The mapping used was randomly initialised and remained static during evolution. The *trivial voting mapping* consisted in taking 3 bits at genotype level to represent one bit a phenotypic level. The latter was set to 1 if the majority of the 3 genotypic bits voted in favour, 0 otherwise. The *standard voting mapping* was a variation of the previous mapping where the set of three genotypic bits encoding for one phenotypic bit can overlap. This means that when a single point mutation takes place, multiple phenotypic bits could simultaneously change. In the *cellular automaton mapping* each of the phenotypic bits was associated with a truth table. Three adjacent bits were used as inputs in the truth table and the corresponding output, which in a cellular automaton would represent the new state of a cell, represented the associated phenotypic bit. Finally, the *random Boolean network (RBN) mapping* was a variation of the previous mapping. The main difference is that the 3 bits can be at any positions, so it is necessary to encode those positions at the genotype level. Shackleton *et al.* noted that the amount of redundancy in the genotype-phenotype map plays a key role in evolution. Moreover, they observed that the standard voting, cellular automaton and RBN mappings were more beneficial than the other two.

Ebner *et al.* [24], [25] extended this investigation. They further analysed the effects of the RBN and the cellular automaton mappings in the context of what they called *phenotype-species* mapping. This type of mapping works in two stages. Firstly, a genotype-phenotype mapping is used to determine the phenotype that corresponds to a genotype. Then, the phenotype-species mapping determines the species to which each phenotype belongs. This phenotype-species mapping is created by randomly distributing the species over the phenotype space. Ebner *et al.* argued that these types of mappings are particularly interesting since they seem to allow neutral networks that are intertwined with a high degree of connectivity. This property allows the finding of more species compared with other types of mappings.

In [26], Knowles and Watson criticised the usefulness of neutrality when added via a mapping function. In particular, they focused their attention on the RBN mapping proposed and studied in [22]–[24] and measured its influence on evolution using the rate of fitness increase. EAs and Hill-Climbers were used on three different problems to compare the performance obtained with and without the RBN mapping. They showed that the performance of these search algorithms was better in the absence of neutrality. Moreover, they suggested that the RBN mapping leads to a random exploration in the search space, so it is difficult to imagine how evolutionary search can gain anything from using this type of mapping.

Rothlauf and Goldberg [27] argued that redundancy is a common element found in any EA and that the effects of redundancy in evolutionary search depend basically on the nature of the redundancy. They identified some properties that are useful to characterise redundant representations: (a) a redundant representation is *uniform* if all phenotypes can be obtained by the same number of genotypes, (b) a representation is *synonymously redundant* if the genotypes that map to the same phenotype are part of a neutral network (i.e., they are close to each other),[3] (c) a redundant representation has high *locality* if neighbouring genotypes map to neighbouring phenotypes and, finally, (d) a redundant representation has high *connectivity* if the number of phenotypes which are accessible from a phenotype by one bit-flip mutation is high. Rothlauf and Goldberg argued that in synonymously redundant representations, genetic operators work well and the landscape is smoother than in non-synonymously redundant representations where the search operators show poor performance. In non-synonymously redundant representations[4] two genotypes representing the same phenotype may be very different from each other and, as a consequence, evolutionary search behaves like random search.

Fonseca and Correia [28] developed two redundant representations using approaches based on mathematical

[3]An example of this type of redundancy is the trivial voting mapping proposed in [22], [23].

[4]Examples of this type of redundancy are the cellular automaton and the RBN mappings described in [22], [23].

tools. They found that some of the properties and analysis of Rothlauf and Goldberg provided in [27] disagreed with their results. In particular, while Rothlauf and Goldberg suggested that when using a synonymously redundant representation the connectivity between phenotypes is not increased, Fonseca and Correia indicated that this is not necessarily true. They reported that, with their proposed representations, the connectivity between phenotypes tended to increase with the amount of redundancy in the encoding. Fonseca and Correia also found that high connectivity can be present even with very little redundancy. Therefore, the belief that large amounts of neutrality must be present to aid evolution [29] should be carefully scrutinised.

As can be seen from the brief survey provided above,[5] there are many contradictory results on neutrality. In the following section, we will present a measure of hardness — the fitness distance correlation — that will later help us explain under what circumstances neutrality can be beneficial in an evolutionary process.

III. Fitness Distance Correlation

Jones [31], [32] suggested that we could consider fitness functions as heuristic functions (in the sense of the term used in classical artificial intelligence). Their outputs could then be interpreted as indicators of the distance between tentative solutions and their nearest global optimum in the search space. If one could express the degree to which the fitness function conveys correct information about such a distance, one would get an idea of how difficult the search is going to be. In order to gather information on the difficulty of a problem one would need to perform two tasks: (a) determining the distance between potential solutions and their nearest global optima, and (b) calculating the fitness of potential solutions. Obviously, step (a) requires that the global optima for a problem be known in advance.

Jones proposed to condense the information gathered in this process using a heuristic measure of problem difficulty called the *fitness distance correlation (fdc)*. 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 of such individuals from the nearest global optimum, *fdc* is given by the correlation coefficient

$$fdc = \frac{C_{FD}}{\sigma_F \cdot \sigma_D}, \tag{1}$$

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. Typically, the n individuals used to compute *fdc* are obtained via some form of random sampling.

Jones [31], [32] suggested that a problem can be classified in one of three classes, depending on the value of *fdc*:

1) *misleading* ($fdc \geq 0.15$), in which fitness tends to increase with the distance from the global optimum;
2) *difficult* ($-0.15 < fdc < 0.15$), for which there is no correlation between fitness and distance; and

[5]A more detailed survey can be found in [30].

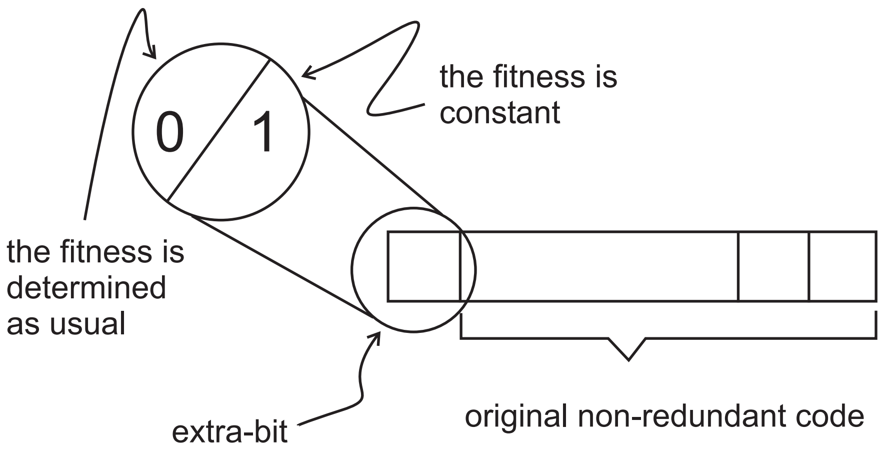

0
1
the fitness is constant
the fitness is determined as usual
extra-bit
original non-redundant code

Fig. 1. Representation used to induce constant neutrality.

3) *easy* ($fdc \leq -0.15$), in which fitness increases as the global optimum approaches.

The interval $[-0.15, 0.15]$ associated with difficult problems was empirically determined.

The fitness distance correlation approach has been successfully used in a wide variety of problems to assess hardness for EAs [33]–[36] and genetic programming systems [12], [37]–[43]. However, there are some known weaknesses of the *fdc* as a measure of problem hardness [44], [45]. Jones himself proposed to use scatter plots of distances *vs.* fitnesses to characterise problems, when *fdc* did not give enough information about the hardness of a problem. Nonetheless, the situations where *fdc* has been shown *not* to be informative are rather artificial.

IV. Constant and Bitwise Neutrality

In the following sub-sections, we will introduce the different forms of neutrality and test problems considered in this paper. To keep things as simple as possible, we will use the simplest possible algorithms — mutation based, binary EAs without crossover — to conduct our studies.

A. The Simplest Form of Neutrality: Constant Neutrality

In the context of binary EAs, the simplest possible definition of neutrality one can imagine is what we will call *constant neutrality*. As illustrated in Figure 1, neutrality is plugged into the original, non-redundant, code by adding an extra bit to the representation. When the bit is set, the fitness of an individual is a pre-fixed constant value. When the bit is not set (i.e., 0), the fitness of the individual is determined by the coding bits as usual. This representation induces a neutral network of constant fitness, identically distributed in the whole search space.

B. Bitwise Neutrality

In this work, we also consider a more natural form of neutrality which we call *bitwise neutrality*. Bitwise neutrality is induced by a genotype-phenotype map, where each phenotypic bit is obtained by transforming n genotypic bits

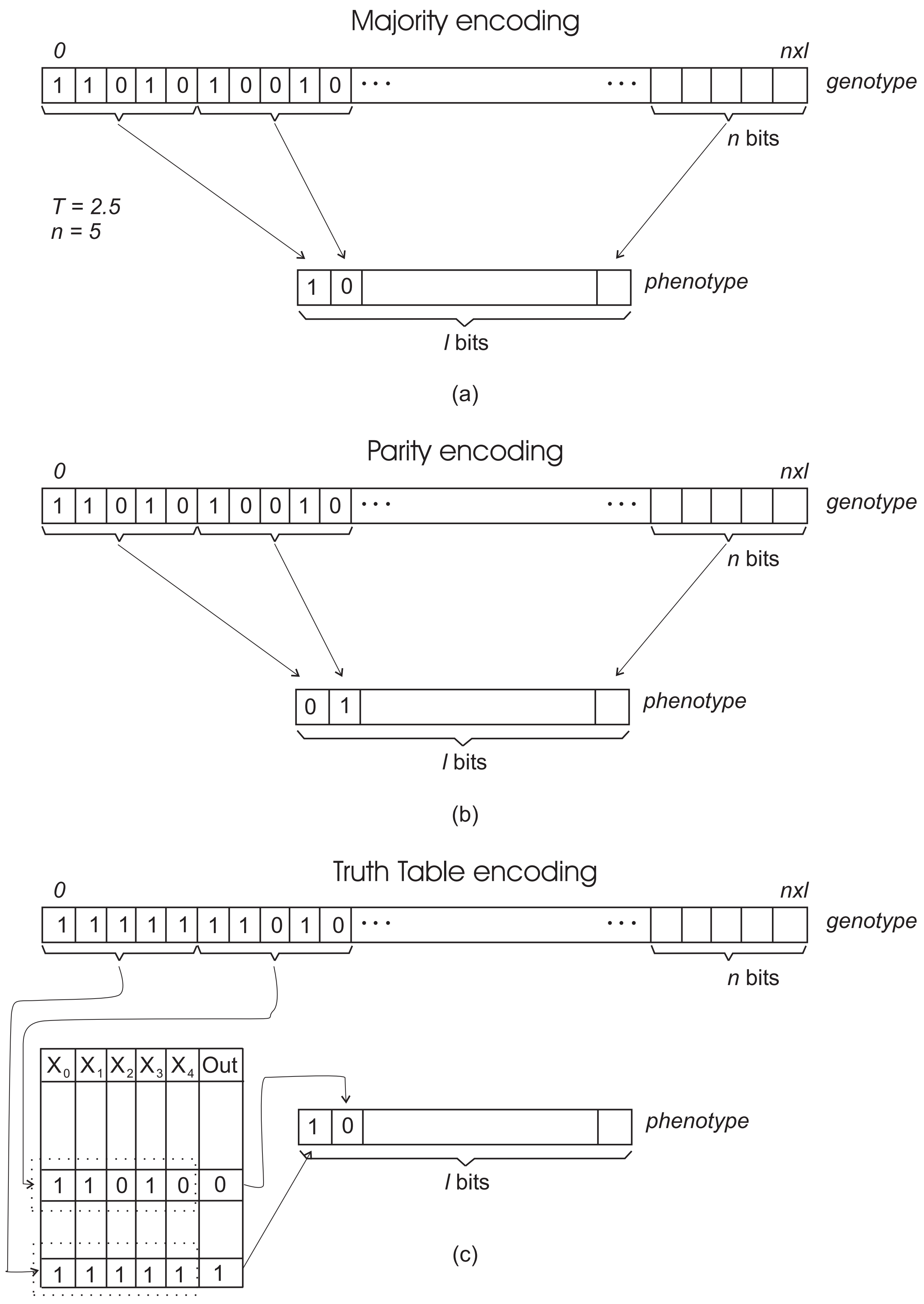


Fig. 2. Three different genotype-phenotype mappings that induce bitwise neutrality: (a) Majority encoding, (b) Parity encoding and (c) Truth table encoding.

via some encoding function. Obviously, whenever $n > 1$, the same phenotype can be obtained from different genotypes, so neutrality is artificially added to the search space.

In particular, we will look at the following three encodings:

1) The *majority* encoding, which works as follows: given n genotypic bits and a user-defined threshold T $(0 \le T \le n)$, if the number of ones in the n bits is greater than or equal to T, then the corresponding phenotypic bit is set to 1, otherwise it is set to 0 (see Figure 2(a)). To avoid biasing the system, we will use $T = n/2$ and n odd, which guarantee that 0s and 1s are treated identically.
2) The *parity* encoding, which works as follows: if the number of ones in n genotypic bits is even, then the corresponding bit of the phenotype is set to 1, otherwise it is set to 0 (see Figure 2(b)).
3) The *truth table* encoding, which works as follows. A truth table is generated where the output associated with each combination of n inputs is randomly set to either 0 or 1, while ensuring the number of 0s and 1s in the table are identical. Each phenotypic bit is determined by using the corresponding n genotypic bits as inputs to the truth table and reading out the associated output (see Figure 2(c)).

C. Test Problems

In order to analyse the effects of the forms of neutrality introduced above, we will use the following problems and classes of problems.

1) OneMax Problem: The problem is to maximise the function:

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

where $x \in \{0,1\}^\ell$ is a binary string of length ℓ and x_i is its i-th element. Naturally, this problem has only one global optimum in $111\cdots111$, and the landscape is unimodal. Seen as a function of unitation (the number of $1s$ in a string), 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.

2) Multimodal Problem Generator: We also use problems generated by the multimodal problem generator presented in [46]–[48]. The idea is to create problem instances with a certain degree of multi-modality.

The generator works as follows. To create a problem with P peaks, P bit strings of length ℓ, which we denote as $Peak_1$, $Peak_2$, ..., $Peak_P$, are randomly generated. To each, a peak height, $Height(Peak_i)$, is assigned. The heights of the peaks are chosen in such a way as to cover an interval $[h, 1]$ (where h is a constant < 1) with P equal-size steps. To evaluate an arbitrary individual, x, it is necessary to first locate the nearest peak in Hamming space, which we denote as

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

where H is the Hamming distance. 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 $Peak_n(x)$, divided by ℓ and scaled by the height of the nearest peak. That is

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

In this problem class, fitness values are in the range $[0, 1]$. The goal is to find the highest peak (i.e., to find a string with fitness 1.0).

The difficulty of problems generated with this technique depends on the number of peaks, the distribution of peaks and, finally, the distributions of peak heights. To carry out our experiments, these parameters have been tuned in such a way to make generated problems much harder than OneMax but easier than the trap problem (described below).

3) Trap Function: The *Trap* function is a deceptive function of unitation [49]–[51] of the following form:

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

where a is the deceptive optimum, b is the global optimum, and u_{min} is the slope-change location. By varying the parameter u_{min}, the relative size of the basins of attraction of the two optima may be varied, thereby making the problem easier or harder.

4) MAX-SAT Problem Class: The Boolean satisfiability problem, also known as SAT, is one of the most studied NP-complete problems (e.g., see [52]–[58]). The target in SAT is to determine whether it is possible to set the variables of a given Boolean expression in such a way to make the expression true. The expression is said to be satisfiable if such an assignment exists.

In SAT, expressions are often represented in conjunctive normal form, i.e., as a conjunction of clauses, where each clause is a disjunction of literals (variables or negated variables). In a version of the problem, called k-SAT, all clauses have exactly k literals. A related problem, known as the Maximum Satisfiability problem, or MAX-SAT, consists in determining the maximum number of clauses of a given Boolean formula that can be satisfied by some assignment. MAX-k-SAT is the maximum satisfiability problem for k-SAT instances. In this paper we use a class of MAX-3-SAT problems as our fourth benchmark. In particular, we focus on problems where the maximum number of satisfiable clauses is equal to the number of clauses in a formula. In other words, the MAX-3-SAT instances we consider are all satisfiable.[6]

We treat MAX-3-SAT as an optimisation problem with the following objective function:

$$f(x) = \sum_{i=1}^{c} S_i(x),$$

[6]The reason for this choice is simple. It is well-known that random 3-SAT instances become harder and harder to satisfy as the clause-to-variable ratio increases (e.g., see [59]). By using satisfiable MAX-3-SAT instances, we can finely control the difficulty of our benchmarks by varying such a ratio.

where $S_i(x)$ is 1 if clause i is satisfied by assignment x and 0 otherwise. A clause is satisfied if at least one of the literals it contains is true. Since our random MAX-3-SAT instances are all satisfiable, we declared a MAX-3-SAT problem as solved as soon as a string x such that $f(x) = c$ was generated by the EA.

V. Fitness Distance Correlation in the Presence of Neutrality

In this section, we will study the forms of neutrality introduced in the previous section using the *fdc*. We do this analytically. To avoid sampling errors, we consider every point in the search space, instead of a random subset.

Naturally, before analysing how the *fdc* for different problems changes in the presence of neutrality, we need to evaluate the *fdc* in its absence.

A. fdc *in the Absence of Neutrality*

For all our test problems, given a search space of binary strings of length ℓ, if the whole search space is sampled in order to compute C_{FD}, we have:

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

where $f(x)$ is the fitness of string x, $d(x)$ is the Hamming distance between string x and its nearest *global* optimum, and $\bar{d}$ and $\bar{f}$ are the averages of $d(x)$ and $f(x)$, respectively, over all strings in the search space. Similar expressions hold for σ_F and σ_D.

Naturally, in problems with a single global optimum (such as OneMax, Trap and the functions created by the multimodal problem generator), we have that $\bar{d} = \frac{\ell}{2}$ in the expressions for C_{FD} and σ_D.

Further simplifications are possible for the OneMax and the Trap problems, since these are functions of unitation. For example, considering that the unitation of the optimal string for these problems is $u_{opt} = \ell$, we can coarse-grain and simplify the calculation of C_{FD} as follows:

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

where, as indicated above, $\bar{d} = \frac{\ell}{2}$ and

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

Similar expressions can be obtained for σ_D and σ_F.

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

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

as one can easily see by noting that $\frac{1}{2^\ell}\binom{\ell}{u}$ is an instance of a binomial distribution function (with success probability $1/2$). Thus,, by the definition of variance, $C_{FD} = -Var[u]$. By similar arguments, one finds $\sigma_D^2 = \sigma_F^2 = \frac{\ell}{4}$, whereby $fdc = -1$, suggesting an easy problem. For Trap functions, instead, whenever $u_{min} \approx \ell/4$, one finds $fdc \approx 1$ [31] indicating hard problems.

B. fdc *in the Presence of Constant Neutrality*

As mentioned in Section IV, constant neutrality is a form of neutrality where an extra bit is added to the genotype. When the bit is set, the individual is on a neutral network and its fitness is a predefined value, f_n, irrespective of the other bits.

In this situation, C_{FD_n} (the subscript n stands for “neutrality”) is given by:

$$C_{FD_n} = \frac{1}{2^{\ell+1}} \sum_{x \in \{0,1\}^{\ell+1}} (f(x) - \bar{f})(d(x) - \bar{d}), \tag{2}$$

where $x = x_0 x_1 \cdots x_{n\ell}$ is a genotype, and x_0 is the “neutrality” bit. Similar expressions hold for σ_D and σ_F. Note that $f(x)$ can be written as

$$f(x) = \begin{cases} f_n & \text{if } x_0 = 1, \\ f_p(x_1, \cdots, x_\ell) & \text{otherwise,} \end{cases}$$

where $f_p(\cdot)$ is the “phenotypic fitness”, i.e., the fitness associated with the coding bits in a genotype. It follows that

$$\bar{f} = \frac{f_n}{2} + \frac{\bar{f}_p}{2},$$

$\bar{f}_p$ being the average fitness in the absence of neutrality.

Note that, assuming all global optima have fitness higher than f_n, the distance of a string $x_0 x_1 \cdots x_\ell$ from the closest optimum is the same as in the absence of neutrality if $x_0 = 0$, while the distance is increased by 1 if $x_0 = 1$ (since the optima are outside the neutral network). Thus,

$$\bar{d} = \frac{1}{2} + \bar{d}_p,$$

d_p being the distance from the closest global optimum in the absence of neutrality.

Substituting these results in Equation (2) we obtain

$$\begin{aligned} C_{FD_n} = & \\ & \frac{1}{2^{\ell+1}} \sum_{y \in \{0,1\}^\ell} (f_n - \frac{f_n}{2} - \frac{\bar{f}_p}{2})(1 + d_p(y) - \bar{d}_p - \frac{1}{2}) \\ + \quad & \frac{1}{2^{\ell+1}} \sum_{y \in \{0,1\}^\ell} (f_p(y) - \frac{f_n}{2} - \frac{\bar{f}_p}{2})(d_p(y) - \bar{d}_p - \frac{1}{2}). \end{aligned}$$

By simplifying and collecting terms appropriately, one can rewrite this as

$$\begin{aligned} C_{FD_n} &= \frac{3}{8}(f_n - \bar{f}_p) \\ &+ \frac{1}{2^{\ell+1}} \sum_{y \in \{0,1\}^\ell} (f_p(y) - \bar{f}_p)(d_p(y) - \bar{d}_p) \\ &= \frac{3}{8}(f_n - \bar{f}_p) + \frac{1}{2} C_{FD}. \end{aligned}$$

Proceeding similarly for σ_F^2 and σ_D^2 we obtain

$$\sigma_{F_n}^2 = \frac{(f_n - \bar{f}_p)^2}{4} + \frac{1}{2}\sigma_F^2$$

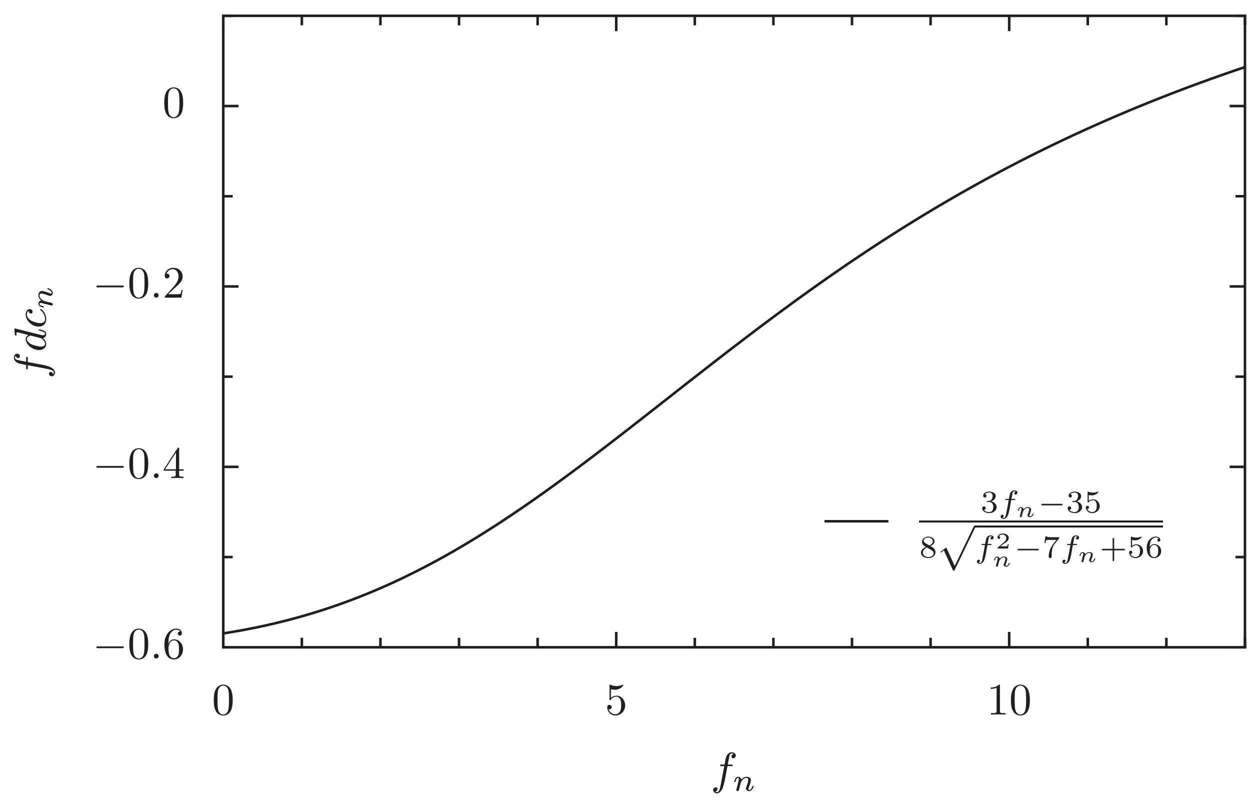


Fig. 3. Fitness distance correlation in OneMax in the presence of constant neutrality as a function of f_n for $\ell = 14$.

and

$$\sigma^2_{D_n} = \frac{1}{2} + \sigma^2_D.$$

Combining the previous results we obtain

$$fdc_n = \frac{\frac{3}{8}(f_n - \bar{f}_p) + \frac{1}{2}C_{FD}}{\sqrt{\frac{(f_n - \bar{f}_p)^2}{4} + \frac{1}{2}\sigma^2_F} \times \sqrt{\frac{1}{2} + \sigma^2_D}}. \tag{3}$$

Equation (3) makes it clear that, in the presence of constant neutrality, *fdc* depends on the difference between the fitness of the neutral network, f_n, and the mean fitness in the absence of neutrality, $\bar{f}_p$. In addition, *fdc* depends on the fitness-distance covariance C_{FD} as well as the variances σ^2_F and σ^2_D in the absence of neutrality.

As an example, let us use Equation (3) to calculate fdc_n for the OneMax problem for $\ell = 14$ (the value of ℓ we will use in the experiments in Section VII). We know that for this problem $C_{FD} = -\frac{\ell}{4}$ and $\sigma^2_F = \sigma^2_D = \frac{\ell}{4}$. Also, $\bar{f}_p = \frac{\ell}{2}$. Since $\ell = 14$, with a little algebra we obtain

$$fdc_n = \frac{3f_n - 35}{8\sqrt{(f_n - 7)^2 + 7}}.$$

Figure 3 shows a plot of this function. We can immediately see from this plot that the addition of neutrality reduces *fdc* (i.e., is expected to make the problem harder) irrespective of the choice of f_n. For particularly high values of f_n, however, fdc_n gets very close to zero, indicating that the problem becomes really difficult.

More generally, from Equation (3), we see that if $f_n = \bar{f}_p$, we have that $fdc_n = \frac{1}{\sqrt{2}} \times \frac{C_{FD}}{\sigma_F \times \sqrt{\frac{1}{2}+\sigma_D^2}}$. This means that if C_{FD} is positive then $fdc_n < \frac{1}{\sqrt{2}} \times \frac{C_{FD}}{\sigma_F \times \sigma_D} = \frac{fdc}{\sqrt{2}}$, while if C_{FD} is negative then $fdc_n > \frac{1}{\sqrt{2}} \times \frac{C_{FD}}{\sigma_F \times \sigma_D} = \frac{fdc}{\sqrt{2}}$. In other words, the addition of constant neutrality with a neutral network of fitness $f_n = \bar{f}_p$ would make easy problems harder and deceptive problems easier.

One might wonder why this would be the case. The reason is that adding a neutral network to the landscape modifies the search process. The search operators will produce individuals on the neutral network (as well as outside it). If the fitness of the neutral network is not too low, selection will use some of these individuals as parents. Since these are all equally good, the search will effectively acquire some of the features of random search. Of course, the part of the population outside the neutral network will be affected and guided by the fitness function. In an easy problem, the guidance will generally be reliable. Therefore, having effectively hybridised the search with random search will simply slow down the process of converging towards high fitness regions of the landscape. However, in deceptive problems, where the information provided by the fitness function does not lead towards the global optimum, the hybridisation is beneficial: by ignoring the guidance of the fitness function at least some of the time, the probability of the searcher stumbling on a good area of the fitness landscapes is increased.

If $f_n \neq f_{\bar{p}}$, as for OneMax, in general we will find that these effects are modulated by the value of f_n. For example, if f_n is high compared with $\bar{f}_p$, the neutral network will act as an attractor for the population. This will make the search more random than in the case $f_n = \bar{f}_p$ considered above. So, in easy problems, we should see a more marked worsening of performance. If, instead, the fitness function was originally deceptive, a high f_n will keep a bigger fraction of the population on the neutral network for longer. This will make it even more probable for the population to escape from traps and locate good optima.

If f_n is low compared with $\bar{f}_p$, selection will avoid using individuals on the neutral network as parents. Consequently, the search does not really become more random than it would be without neutrality. In other words, a neutral network with a low f_n does not really change hard problems into easier ones nor does it change easy problems into harder ones. However, in both cases the search operators (particularly mutation) may produce individuals on the neutral network. These individuals represent wasted samples. So, constant neutrality with a low f_n cannot provide any benefit at all.

One other aspect should be considered. In the presence of constant neutrality, the landscape is divided into two areas of identical sizes: the neutral network and the rest of the search space. For bit strings of length ℓ, there are 2^ℓ points in each region. However, there is still the same number of global optima. This means that the addition of constant neutrality comes at a cost since the size of the search space has been expanded without correspondingly expanding the solution space. Thus, we should expect to see benefits of constant neutrality (e.g., improved performance) only when neutrality modifies the search bias of an algorithm-problem pair in such a way as to make the sampling of the global optimum significantly more likely than without this form of neutrality. If this does not happen, or worse, if the original search bias is modified in such a way as to make it harder to reach the global optimum, then we should expect constant neutrality to be deleterious. These considerations apply also to other forms of neutrality wherever neutrality alters unfavourably the proportion of solutions in the search space.

C. fdc *in the Presence of Bitwise Neutrality*

As mentioned in Section IV, another form of neutrality that is considered here is the one where each phenotypic bit is encoded using n genotypic bits. In this situation, C_{FD_n} is given by:

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

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

$$f(x) = f_p(g(x^{(1)}), g(x^{(2)}), \cdots, g(x^{(\ell)})) \tag{4}$$

where $x^{(k)} = x_{(k-1)n+1} \cdots x_{kn}$ is a sub-string of x, g is one of our encoding functions (e.g., Majority or Parity), and $f_p(y)$ is the phenotypic fitness ($y \in \{0,1\}^\ell$).

Let us define two sets: $X_n = \{x \in \{0,1\}^n : g(x) = 1\}$ and $\bar{X}_n = \{x \in \{0,1\}^n : g(x) = 0\}$. In what follows we require that the 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 IV. So, $|X_n| = |\bar{X}_n| = 2^{n-1}$.

If one knows the function f_p and the location of its global optima, one can simplify the expressions of C_{FD_n}, σ_{D_n} and σ_{F_n}. This results in a formula for fdc_n where there is an explicit dependency on the number of bits, n, used in the encoding functions which induce bitwise neutrality. One can then see under what circumstances bitwise neutrality makes problems easier or harder (note that when $n = 1$ there is no added neutrality in the encoding).

The calculations involved here are doable but they are rather laborious. For this reason, in the following section we will illustrate the process using the simplest of our test functions: OneMax.

D. fdc *for OneMax with Bitwise Neutrality: General Results*

For the OneMax function, the phenotypic fitness of a bit string $y_1 ... y_\ell$ is $f_p(y_1, ... y_\ell) = \sum_i y_i$. Thus, from Equation (4) we obtain

$$f(x) = \sum_i g(x^{(i)}). \tag{5}$$

To compute *fdc* we use a result originally derived by Jones [31, 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 to Equation (5) because g can be interpreted as the fitness function of an n-bit problem which is concatenated ℓ times to form an $\ell \times n$ bit problem with fitness function $f(x)$. Therefore, *fdc* for OneMax can be computed for 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 much simpler than the original.

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

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

Using this result in the computation of σ_F^2, we obtain

$$\sigma_F^2 = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (g(x) - \bar{g})^2 = \frac{1}{4},$$

which, again, is valid for all encodings.

By definition

$$\bar{d} = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} H(x, N(x)), \tag{6}$$

where $N(x)$ is the global optimum of g nearest to x and H is the Hamming distance. Because g can only take two values, 0 and 1, all elements of X_n are global optima of g. So, if $x \in X_n$, then $x = N(x)$ and $H(x, N(x)) = 0$. As a result, Equation (6) simplifies to

$$\bar{d} = \frac{1}{2^n} \sum_{x \in \bar{X}_n} H(x, N(x)). \tag{7}$$

If the definition of Hamming distance is extended to sets via the definition $H(x, S) = \min_{y \in S} H(x, y)$, Equation (7) becomes

$$\bar{d} = \frac{1}{2^n} \sum_{x \in \bar{X}_n} H(x, X_n) = \frac{1}{2} E[H(x, X_n) | x \in \bar{X}_n], \tag{8}$$

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.

Let us now compute σ_D^2. We have

$$\begin{aligned}
\sigma_D^2 &= \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (H(x, N(x)) - \bar{d})^2 \\
&= \frac{1}{2^n} \sum_{x \in X_n} \left(0 - \bar{d}\right)^2 \\
&+ \frac{1}{2^n} \sum_{x \in \bar{X}_n} \left(H(x, X_n) - \bar{d}\right)^2 \\
&= \frac{1}{2} \left(\bar{d}^2 + E\Big[\left(H(x, X_n) - \bar{d}\right)^2 \Big| x \in \bar{X}_n \Big]\right) \\
&= \frac{1}{2} E\Big[H(x, X_n)^2 \Big| x \in \bar{X}_n \Big] \\
&- \frac{1}{4} \left(E\Big[H(x, X_n) \Big| x \in \bar{X}_n \Big] \right)^2.
\end{aligned}$$

Finally, we compute

$$\begin{aligned}
C_{FD} &= \frac{1}{2^n}\sum_{x\in\{0,1\}^n}(g(x)-\bar{g}\,)(H(x,N(x))-\bar{d}\,)\\
&= \frac{1}{2^n}\sum_{x\in X_n}\left(1-\frac{1}{2}\right)\left(0-\bar{d}\,\right)\\
&+ \frac{1}{2^n}\sum_{x\in \bar{X}_n}\left(0-\frac{1}{2}\right)(H(x,X_n)-\bar{d}\,)\\
&= \frac{1}{2^{n+1}}\sum_{x\in \bar{X}_n}H(x,X_n)\\
&= -\frac{1}{4}E[H(x,N(x))|x\in\bar{X}_n]\\
&= -\frac{\bar{d}}{2}
\end{aligned}$$

In the following subsections, these generic results are applied to the three encoding functions presented previously: Parity, Truth Table and Majority.

E. fdc *for OneMax under Parity Bitwise Neutrality*

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 $E[H(x,X_n)|x\in\bar{X}_n]=1$. Thus, from Equation (8) one obtains $\bar{d}=\frac{1}{2}$. It follows that $C_{FD}=-\frac{1}{4}$. We also have $E\big[H(x,X_n)^2\big|x\in\bar{X}_n\big]=1$. So, $\sigma_D^2=\frac{1}{4}$.

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

$$fdc=\frac{-\frac{1}{4}}{\sqrt{\frac{1}{4}}\sqrt{\frac{1}{4}}}=-1.$$

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. This was expected, since the parity encoding is a form of trivial neutrality [20] (see Section II). So, the difficulty of OneMax should be unaffected by this form of neutrality.

F. fdc *for OneMax under Truth Table Bitwise Neutrality*

Let us now consider the Truth Table encoding. In order to apply Equation (8), we need to compute $E[H(x,X_n)|x\in\bar{X}_n]$. To do this, we treat $H(x,X_n)$ as a stochastic variable. We want to compute the probability, $p(d)$, that $H(x,X_n)=d$ for a randomly chosen $x\in\bar{X}_n$. Then, making use of the definition of the expected value, we want to compute

$$E[H(x,X_n)|x\in\bar{X}_n]=\sum_{d=1}^{n}d\cdot p(d). \tag{9}$$

We start by considering the case $d=1$. 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 with success probability $\frac{1}{2}$, we have that

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

Thus, 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. These two events are treated as independent.[7] Obviously, the probability that none of the Hamming-1 neighbours of x is a member of X_n is simply $1 - p(1) = \left(\frac{1}{2}\right)^n$. The probability of at least one of its Hamming-2 neighbours being in X_n is the complement of the probability that none of the Hamming-2 neighbours 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 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

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

Generalising the calculation we get

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

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 this means that typically only $p(1)$ and $p(2)$ have any relevance in the calculation in Equation (9).

As a result, for sufficiently large n we can approximate

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

So, for the Truth Table encoding, we have

$$\bar{d} \approx \frac{1}{2} + 2^{-n-1}, \quad \text{whereby} \quad C_{FD} = -\frac{1}{4} - 2^{-n-2}.$$

Using similar approximations, we find that

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

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

From this, it follows that

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

Therefore, the fitness distance correlation for OneMax under the Truth Table encoding can be approximated as

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

This equation has been derived using approximations that are valid for sufficiently large n. For such values of n, the constant terms in the equation will tend to dominate and effectively $fdc \approx -1$. This means that the Truth Table encoding induces a form of neutrality which, for sufficiently large n, 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.

G. fdc *for OneMax under Majority Bitwise Neutrality*

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 where $T=n/2$ 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.[8] That is, one can naturally describe $\bar{X}_n$ by saying that it 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

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

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 $0.63+0.37\sqrt{n}$. Thus, we have

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

and

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

[8]The operation $\lfloor T\rfloor$ returns the largest integer not bigger than T, while $\lceil T\rceil$ returns the smallest integer not smaller than T.

Using a similar approach, we compute

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

whereby

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

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

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

This equation makes it clear that, in this case, there is a much more marked effect of the encoding on the difficulty of a problem: the *fdc* progressively increases (from the original value of -1) as 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.

H. Lessons for other problems

Naturally, *fdc* could be computed also for the Multimodal problem generator, the Trap function and any given MAX-SAT problem in the presence of bitwise neutrality. Unfortunately, for these functions one cannot not use Jones' "trick" [31, Appendix D] to simplify the calculations. This makes the derivation of theoretical results much more complex. However, based on what we have learnt from the results on constant neutrality, from our results with bitwise neutrality and OneMax, and from the theory in [20], it is easy to understand that the Parity and Truth Table encodings will have a limited influence on the *fdc* of the Trap, Multimodal and MAX-SAT functions. However, we should expect the Majority encoding to change the *fdc* (and potentially the difficulty) of these problems significantly.

VI. 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 the difficulty of a problem is decreased, increased or left unaltered by neutrality. However, fitness landscapes and *fdc* effectively neglect to model the fact that the precise distribution of mutants may have an important effect on search behaviour and performance. For example, *fdc* remains the same irrespective of the mutation probability p_m.

Thus, to evaluate the benefits and drawbacks of neutrality it is also important to understand what effects different types of neutral encodings have on the way the search proceeds. In particular, we want to understand how genotypic mutations are related to phenotypic mutations, since only phenotypic changes can lead to fitness changes. To do so, the notion of *phenotypic mutation rate* will be used.

A. Constant Neutrality

In the case of constant neutrality, the genotype-phenotype mapping is not fully specified. This is because we directly associate a fitness f_n to all bit strings of the form $x_0x_1...x_\ell$ with $x_0 = 1$, without really going through a process of transformation of the genotype into a phenotype. Only when $x_0 = 0$ the genotype-phenotype mapping is specified: this is a simple transformation, where the phenotype is directly determined by the genotypic bits $x_1...x_\ell$. However, if we imagine, for simplicity, that all strings on the neutral network represent the same phenotype, we have a fully specified genotype-phenotype map for constant neutrality. This allows us to define the notion of phenotypic mutation rate for this representation. To further simplify our treatment we will further assume that the "neutral" phenotype is different from all other phenotypes.

Let p_m be the mutation rate and let p_p be the probability of a phenotype change when the genotype is hit by a mutation in the absence of neutrality. In general, p_p is a monotonically increasing function of p_m (more on this below). Let $p(x_0 = 1)$ and $p(x_0 = 0)$ represent the probability of selection for individuals on the neutral network and outside the neutral network, respectively.

Let us consider the possible ways in which the bit x_0 can be modified by a mutation and what consequences this has on the phenotype represented by a string $x_0x_1...x_\ell$:

(1) If a parent string has $x_0 = 1$ and the bit x_0 is not mutated, irrespective of how many mutations will hit the remaining bits in the string, there cannot be a phenotypic change (the offspring will still be on the neutral network as its parent). If instead x_0 is mutated into a 0, which happens with probability p_m, then there is always a phenotypic change.

(2) If a parent string has $x_0 = 0$ and the bit x_0 is not mutated, which happens with probability $1 - p_m$, then the mutations on bits $x_1...x_\ell$ determine whether there is a phenotypic change. This will occur with probability p_p. If instead x_0 is mutated into a 1, which happens with probability p_m, then, again, there is always a phenotypic change.

Naturally, case (1) applies only to a proportion $p(x_0 = 1)$ of mutations, while case (2) applies to a proportion $p(x_0 = 0)$.

By properly combining these probabilities, one finds that the probability of a phenotypic mutation p_{p_n} in the presence of constant neutrality is

$$\begin{aligned} p_{p_n} &= p(x_0 = 1)p_m + p(x_0 = 0)[(1 - p_m)p_p + p_m] \\ &= p_m + p(x_0 = 0)(1 - p_m)p_p \qquad (10) \\ &\approx p(x_0 = 0)p_p, \end{aligned}$$

where we used the property $p(x_0 = 1) + p(x_0 = 0) = 1$ and we assumed that p_m is small. In other words, we have that with constant neutrality the probability of a phenotypic mutation is proportional to the probability of a phenotypic mutation observed in the absence of neutrality. The proportionality factor — the selection probability for strings outside the neutral network — is not fixed. It depends on how attractive individuals in the neutral network

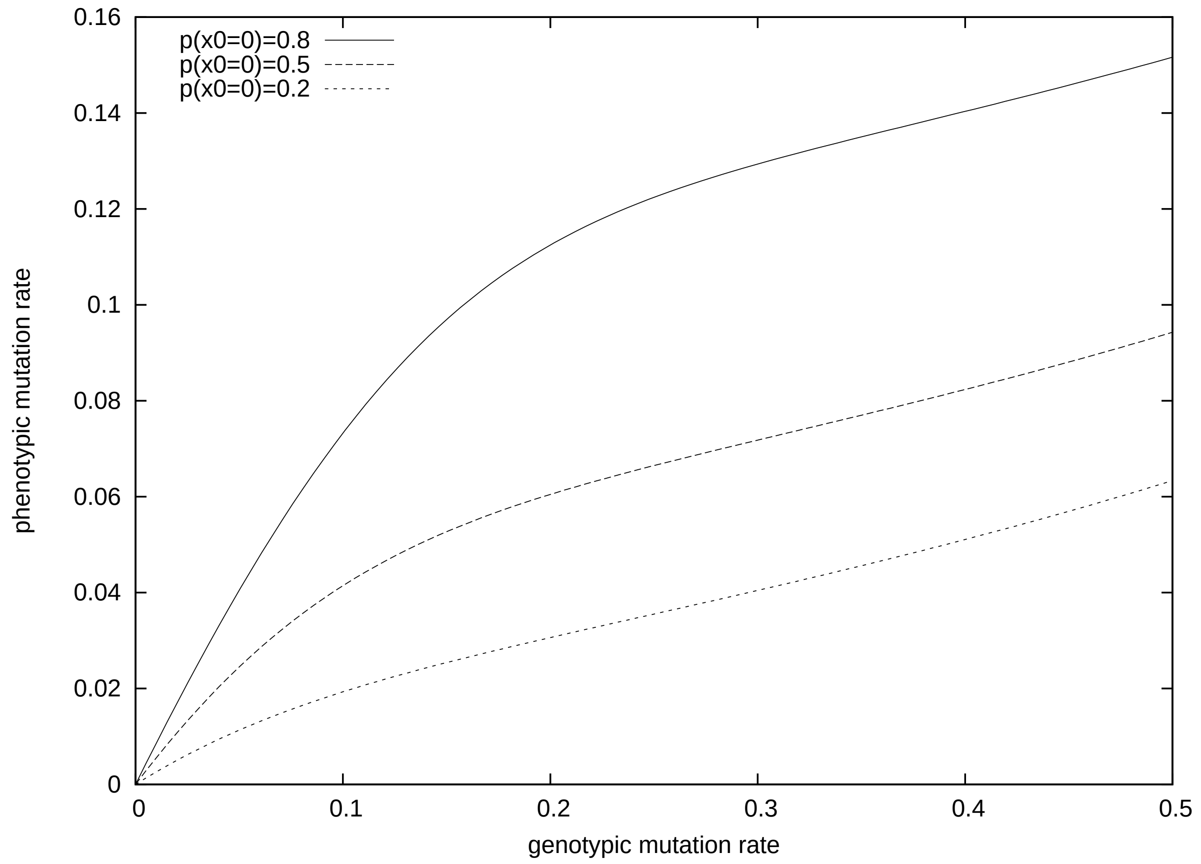


Fig. 4. Effective phenotypic mutation rate for invertible fitness functions as a function of the genotypic mutation rate for strings of length $\ell = 14$ and different values of the selection probability for strings outside the neutral network induced by constant neutrality.

appear to selection. If f_n is high, we should expect to see a bigger proportion of the population on the neutral network than if f_n is low. If f_n is high, we should expect the selection probability $p(x_0 = 0)$ and, correspondingly, the probability of a phenotypic mutation to be smaller than if f_n is low, and *vice versa*.

Note that the exact expression of the probability of a phenotypic mutation p_p depends on the problem. If the fitness function f_p is invertible, each string $x_1 \cdots x_\ell$ has a unique fitness associated to it. Therefore, effectively any mutation hitting bits x_1 to x_ℓ causes a phenotypic mutation. In these cases, p_p is the complement of the probability that none of the bits in $x_1 \cdots x_\ell$ are mutated, i.e., $p_p = 1 - (1 - p_m)^\ell$.

Let us define the *effective phenotypic mutation rate*, p_{m_p}, for constant neutrality as a quantity such that the probability of a phenotypic mutation in the left-hand-side of Equation (10) can be rewritten as $p_{p_n} = 1-(1-p_{m_p})^\ell$. For invertible fitness functions, this is exactly the same form as p_p. For such functions, substituting the expressions for p_p and p_{p_n} in Equation (10) and solving for the effective phenotypic mutation rate yields

$$\begin{aligned} p_{m_p} = & \\ & 1-[p(x_0=0)(1-p_m)^\ell + p(x_0=0)p_m + 1 \\ & -p(x_0=0)((1-p_m)^\ell p_m) - p(x_0=0) - p_m]^{1/\ell}. \end{aligned} \tag{11}$$

This equation allows us to get a feel for how constant neutrality alters the effects of the mutation operator. Figure 4 illustrates the relationship, indicating how, as expected, constant neutrality reduces the effective mutation rate used in the search.

B. Bitwise Neutrality

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

$$p_{m_p} = \sum_{i=1,3,5,\ldots} \binom{n}{i} p_m^i (1-p_m)^{n-i}.$$

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:

$$p_{m_p} = \frac{1-(1-p_m)^n}{2}.$$

This is because there is the potential for a change in a phenotypic bit whenever the row is changed from which the output in the truth table is read. This happens if at least one genotypic mutation takes place (hence the factor $1-(1-p_m)^n$). 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 is more difficult. However, obtaining numerical estimates for these is very easy. This can be done by generating genotypic mutants of groups of n bits using a particular genotypic mutation rate, and recording how frequently the mutants are in a different majority class than the original configuration.

Table I shows the phenotypic mutation rates corresponding to the mutation rates $p_m = 0.01$, 0.06 and 0.1 for Parity, Truth Table and Majority. In the case of Majority, the figures are estimated by generating 10,000 mutants starting from a uniform random population.

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**, $\underline{\text{underlined}}$, doubly $\underline{\underline{\text{underlined}}}$ and underlined with a wavy line in the table. Note that the Parity and Truth Table encodings (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 the phenotypic mutation rates also match, we should expect to see similar performance under these two encodings.

TABLE I

PHENOTYPIC MUTATION RATES CORRESPONDING TO DIFFERENT GENOTYPIC MUTATION RATES FOR DIFFERENT FORMS OF BITWISE NEUTRALITY.

Type of redundancy	$p_m = 0.01$	$p_m = 0.06$	$p_m = 0.1$
Parity (n bits = 5)	0.0480	$\underline{\underline{0.2361}}$	$\utilde{0.3362}$
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	$\utilde{0.3202}$	0.4161
Truth Table (n bits = 5)	0.0245	0.1331	$\underline{0.2048}$
Truth Table (n bits = 6)	0.0293	0.1551	$\underline{\underline{0.2343}}$
Truth Table (n bits = 7)	0.0340	0.1758	0.2609
Truth Table (n bits = 8)	0.0386	$\underline{0.1952}$	**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

VII. EXPERIMENTAL RESULTS

In Section IV-C, we presented the problems used to conduct our experiments. In the experiments with OneMax, Trap and the multimodal landscapes we used chromosomes of length $\ell = 14$. In the MAX-3-SAT problem domain the size of the chromosomes, ℓ, was determined by the number of variables, v. Thus, we used $\ell = 7$, $\ell = 10$ and $\ell = 14$.

For the multimodal landscape, we set $P = 400$ (i.e., there are 400 peaks). These were distributed in such a way to give the problem deceptive features. Specifically, the highest peak was at position $11 \cdots 1$, the second highest peak was at position $00 \cdots 0$, and the remaining peaks were randomly distributed. This last feature makes the problem easier than the trap function.

For the trap function, the following parameters were used: $u_{min} = 13$, $a = 39$, $b = 40$. Figure 5 depicts this trap function.

We created MAX-3-SAT instances by randomly constructing clauses (disallowing repeated literals). To build problems of varying difficulty we varied the ratio between the number of clauses and variables in the range 2 (easy) to 6 (very hard). We ensured that all SAT instances were satisfiable by brute force testing of all possible assignments or, for the larger instances, by using the latest version of WalkSat [60]. We considered problems including between $c = 14$ and $c = 84$ clauses.

The experiments were conducted using an EA with selection, bit-flip mutation and no crossover. Runs were stopped when the maximum number of generations was reached. For the OneMax, Trap and multimodal problems fitness proportionate selection was used; the other parameters of our runs are given in Table II. For MAX-3-SAT we used a form of EA which is more common in practical applications: a steady-state EA with tournament selection (which provides good control on selection pressure); other run parameters will be provided in Section VII-B4.

For the OneMax, Trap and multimodal problems, a sample size of 4,000 has been used to calculate *fdc*. For

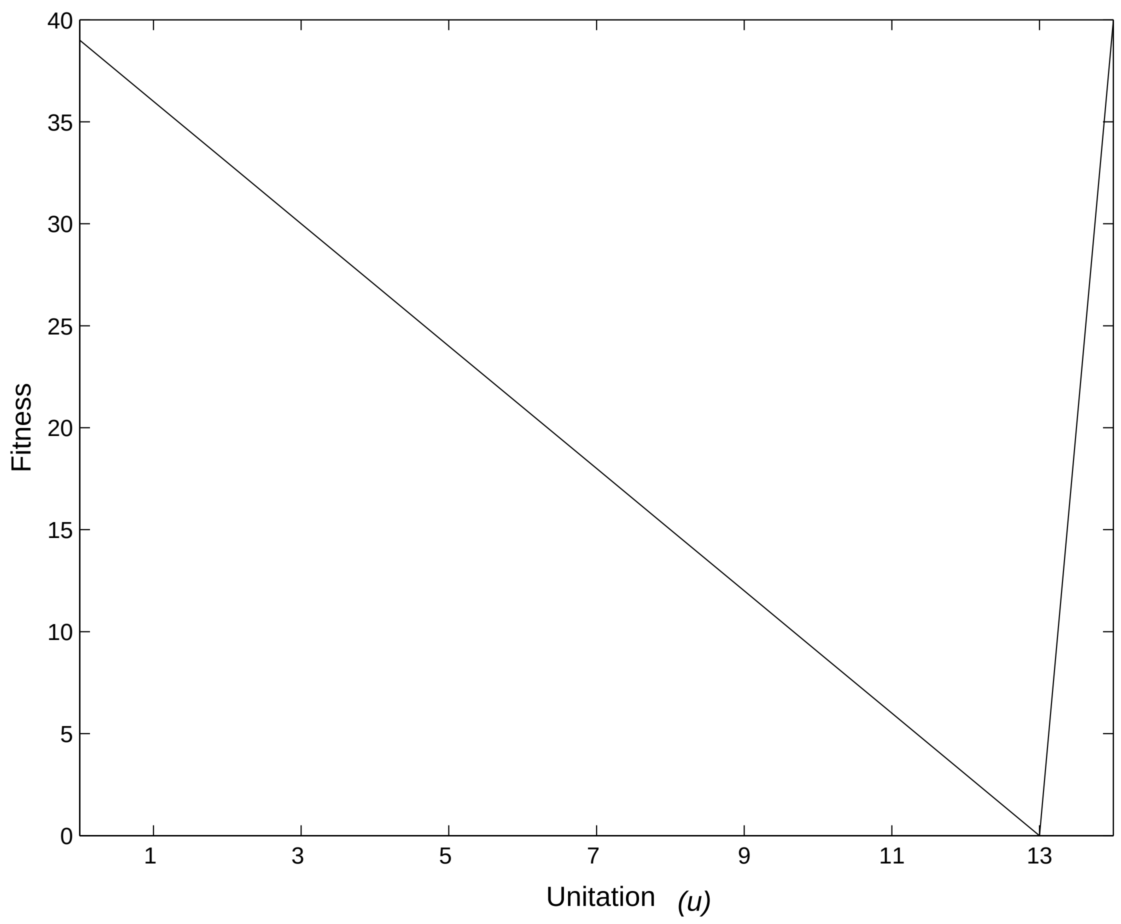


Fig. 5. The trap function used in our experiments ($u_{min} = 13$, $a = 39$, $b = 40$).

MAX-SAT the sample size was the size of the search space.

A. Constant Neutrality

To empirically test the effects of constant neutrality and corroborate the theory presented in Sections V-B and VI-A, we used three problems: OneMax, where neutrality is always expected to be detrimental, the trap function, where we expect neutrality to aid evolution, and MAX-SAT which we suspected to be half-way between the first two.

1) OneMax Problem: Let us start by looking at the results of the experimentation with the OneMax problem. Table III reports the *fdc*, the number of generations required to reach the optimum solution and the percentage of successes for the OneMax problem with and without constant neutrality. As one can see, the *fdc* correlates with the difficulty of the problem in terms of percentage of successes and/or number of generations required to find a solution. For example, we see a significant decrease in performance associated with the large change to *fdc* resulting from the introduction of constant neutrality. Also, we see that as f_n increases, *fdc* increases and so does

TABLE II

PARAMETERS USED FOR THE EXPERIMENTS USING CONSTANT AND BITWISE NEUTRALITY FOR THE ONEMAX, TRAP AND MULTIMODAL PROBLEMS.

Parameter	*Value*
Length of the genome	14
Population Size	80
Generations	100
Mutation Rate (per bit)	0.01, 0.06, 0.1
Generation gap	1
Independent Runs	1,000

TABLE III

PERFORMANCE OF A MUTATION-BASED EA ON THE ONEMAX PROBLEM USING CONSTANT NEUTRALITY.

	fdc	$p_m = 0.01$		$p_m = 0.06$		$p_m = 0.1$	
		Avr. Gen	% Suc.	Avr. Gen	% Suc.	Avr. Gen	% Suc.
No neutrality	-1.0	21.11	100.0	14.39	100.0	16.47	100.0
$f_n = 11$	-0.1645	38.12	63.0	29.60	99.0	31.81	99.1
$f_n = 12$	-0.0914	38.79	19.8	46.15	68.9	44.77	82.1
$f_n = 13$	-0.0396	27.47	2.0	48.63	12.3	43.14	13.1

the difficulty of the problem.

These results are not surprising, as we argued previously. In the case considered here ($\ell = 14$) the maximum achievable fitness is 14, so a neutral network with fitness $f_n = 13$ turns the search of a mutation-only EA into a set of parallel random walks. This is why performance decreases so much. On the contrary, when the fitness of the neutral network is lower, e.g., $f_n = 11$, the original character of the search is maintained. However, the number of generations required to find a solution is increased (in fact, by a factor of about 2) w.r.t. the no-neutrality case.

The experimental results in Table III are in agreement with the theory presented in Section V-B: the problem is getting harder in the presence of constant neutrality and the higher f_n the lower the performance of the EA. Note also that the values of *fdc* reported in Table III are very similar to the corresponding predictions obtained via Equation (3) (e.g., see Figure 3).

If we compare the performance of different representations when p_m is varied in Table III, we can see a reduction in performance in the presence of the unusually low mutation rate of $p_m = 0.01$. In the absence of neutrality, the success rate is still 100%, thus the problem remains very easy although we can see a 50% increase in the average number of generations required to find the optimum. When constant neutrality is used, however, the drop in performance seems to be modulated by p_m. By definition, *fdc* cannot predict this type of effect (see Section III).

Note also that, while *fdc* changes by a very small amount as we go from $f_n = 12$ to $f_n = 13$, the EA's success rate drops significantly. The magnitude of change in *fdc* does not seem to correlate well with the magnitude of the

TABLE IV

PERFORMANCE OF A MUTATION-BASED EA ON THE TRAP FUNCTION WITH AND WITHOUT CONSTANT NEUTRALITY.

	fdc	$p_m = 0.01$		$p_m = 0.06$		$p_m = 0.1$	
		Avr. Gen	% Suc.	Avr. Gen	% Suc.	Avr. Gen	% Suc.
No neutrality	1.0	1	0.6	1	0.6	1	0.6
$f_n = 30$	0.5909	-	0.0	38.37	1.6	39.91	3.6
$f_n = 38$	0.4908	28.66	0.3	44.34	2.9	51.33	5.3

performance drop.

Focusing on the phenotypic mutation rates can give us some explanations. If f_n is relatively low, after a while selection will start neglecting the neutral network, i.e., $p(x_0 = 0)$ will start to increase towards 1. So, the probability of phenotypic mutations, p_{p_n} in Equation 10, will tend to approach p_p. However, for higher values of f_n this happens later in the search. In particular, for $f_n = 13$ all points in the search space (except the global optimum) are at or below the fitness of the neutral network. Thus, $p(x_0 = 0)$ may be small for most of the time, effectively reducing the number of genotypic mutations that can generate new phenotypes. The more we reduce p_m, the more the effect becomes important: if the probability of a phenotypic mutation becomes very small, then most offspring will be identical to their parent, thereby slowing down the search. With $p_m = 0.01$ the search is already slowed down significantly even in the absence of neutrality (only one in about 7 mutants actually being different from their parents). The composition of this effect with the modulation of the probability of phenotypic mutations due to f_n explains why results are so poor for $p_m = 0.01$ and high f_n.

2) The Trap Function: Table IV reports results for the Trap function with and without constant neutrality.

As expected based on the predictions we made in Section V-B, the situation is reversed with respect to the case of the OneMax problem. The addition of constant neutrality here is beneficial for relatively high values of the fitness of the neutral network f_n (the maximum fitness value for the problem is 40).[9] Furthermore, we see that the higher the value of f_n, the less difficult the problem.

Again, if we look at what happens as p_m is varied, we see that the success rate reduces as p_m is reduced and drops to almost zero for $p_m = 0.01$. Again, these findings could not be explained by simply considering the *fdc*. As for the case of OneMax, however, the reduction is clearly an effect of the probability of phenotypic mutations being reduced in the presence of constant neutrality.

3) MAX-3-SAT: As shown in Figure 6, in the case of MAX-SAT instances, the *fdc* increases with the clause-to-variables ratio (i.e., with the difficulty of the problem) irrespective of whether constant neutrality is used or not. However, we see that the addition of constant neutrality increases the *fdc* suggesting that this form of neutrality is harmful in this problem. Also, the figure clearly shows that setting the fitness of the neutral network, f_n, to a high value changes the *fdc* dramatically. Instead, *fdc* is much less affected when $f_n = c/2$.

[9] As we discussed in Section V-B, constant neutrality with low f_n cannot be beneficial.

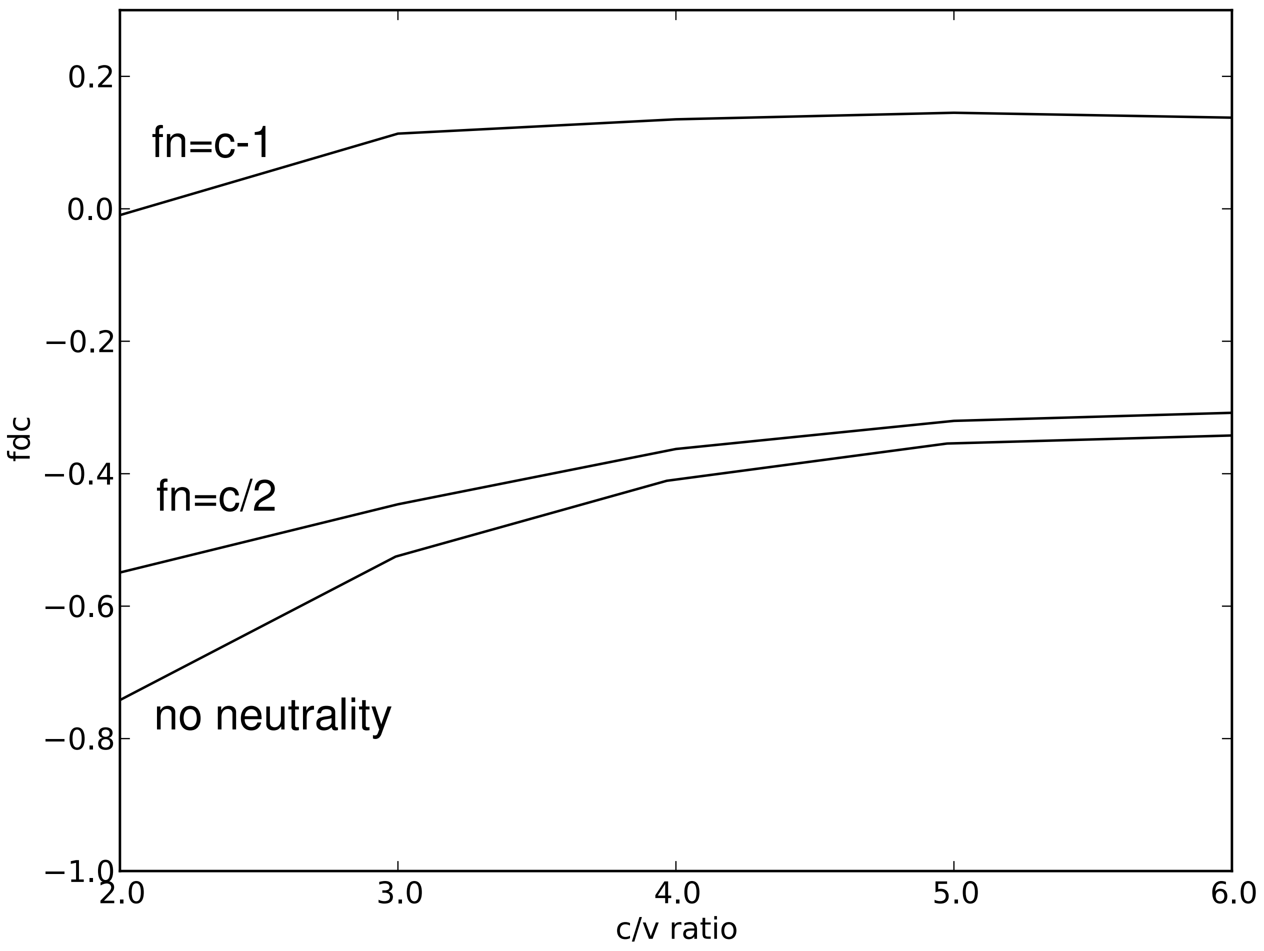


Fig. 6. Fitness distance correlation in MAX-3-SAT problems of increasing complexity in the absence of neutrality and in the presence of constant neutrality, with two different values of f_n: $f_n = c/2$ where c is the number of clauses and $f_n = c - 1$ (c is also the fitness of the global optimum in our SAT problems). The data are averages over 100 random satisfiable 3-SAT instances.

Figure 7 reports the success rates of our mutation-based EA on MAX-3-SAT problems with 14 variables as a function of the problem difficulty, the genotypic mutation rate and f_n. Success rates for the no-neutrality case are also provided for reference.

The figure confirms that, as suggested by the *fdc* analysis (and prior knowledge on SAT), problems get harder, i.e., success rates decrease, as the clause-to-variable ratio increases. Note that success rates are generally reduced in the presence of constant neutrality, confirming the prediction of the analysis of *fdc*. Also, we find that constant neutrality with $f_n = c/2$ is associated with a slightly higher performance than in the case $f_n = c-1$, but results are very close. Even though we report averages over 1,000 runs, differences are unlikely to be statistically significant. Given the significant differences shown by the two encodings in terms of *fdc* (see Figure 6) , one might have expected to see correspondingly big differences in performance. However, there is no evidence of this being the case.

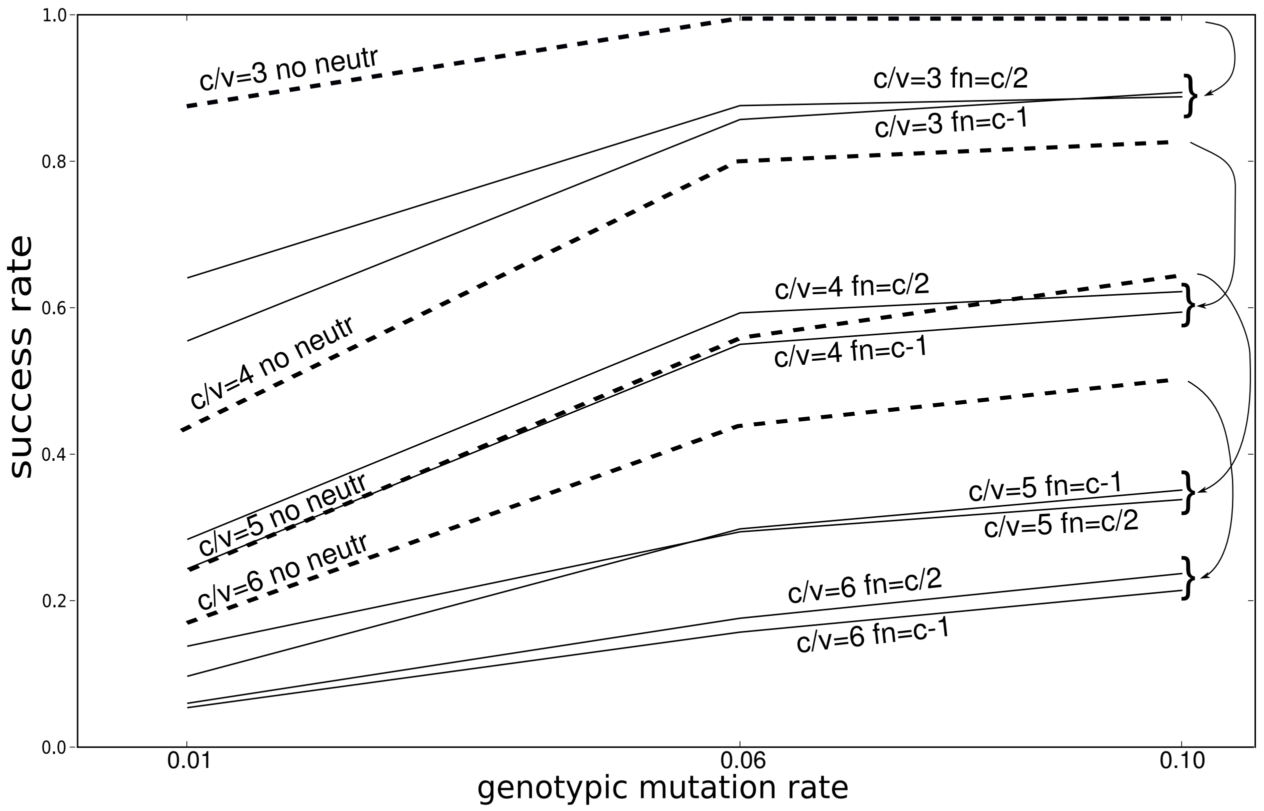


Fig. 7. Plots of the success rate of a mutation-based EA on MAX-3-SAT problems with 14 variables as a function of the problem difficulty, the genotypic mutation rate and the fitness of the neutral network induced by constant neutrality (solid lines). The correspondence between these and success rates in the absence of neutrality (dashed lines) is indicated by the curved arrows on the right.

The reason why performance figures are so close is, again, very much related to the probability of phenotypic mutations and its dependency on selection. We performed some theoretical calculations and some associated numerical simulations (not reported) to identify the fixed-points for the probability of selecting individuals outside the neutral network, $p(x_0 = 0)$. Results show that with tournaments of size of 2, $p(x_0 = 0)$ varies significantly with f_n (in addition to varying with p_m) and that such variation implies that $f_n = c - 1$ has a higher probability of producing phenotypic mutations. This increased search vigour is responsible for the performance of constant neutrality with $f_n = c - 1$ not being too far from the performance of $f_n = c/2$ despite the fitness landscape in the former providing much less guidance than the fitness landscape in the latter (as indicated by the *fdc*).

B. Bitwise Neutrality

To empirically study the effects of bitwise neutrality and verify the theory presented in Sections V-C and VI-B, we used all four test problems described in Section IV-C.

1) OneMax Problem: Table V (second column) reports the *fdc* for OneMax for a representation without neutrality and for various forms of neutral encodings (i.e., Parity, Majority, Truth Table). As predicted in Sections V-D

TABLE V

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	1
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

TABLE VI

PERFORMANCE OF A MUTATION-BASED EA ON THE ONEMAX PROBLEM. PAIRS OF NUMBERS IN **boldface**, $\underline{\text{UNDERLINED}}$, DOUBLY $\underline{\underline{\text{UNDERLINED}}}$ AND UNDERLINED WITH A WAVY LINE REPRESENT SITUATIONS WITH ALMOST IDENTICAL PHENOTYPIC MUTATION RATES.

Type of	$p_m = 0.01$		$p_m = 0.06$		$p_m = 0.1$	
redundancy	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	$\underline{\underline{90.1}}$	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	$\underline{95.0}$
Truth Table ($n = 6$)	16.89	100	22.87	99.4	33.14	$\underline{\underline{90.5}}$
Truth Table ($n = 7$)	15.89	100	24.41	97.5	35.49	84.5
Truth Table ($n = 8$)	15.01	100	28.16	$\underline{97.4}$	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

and V-E, the Parity and Truth Table encodings leave the *fdc* unchanged w.r.t. whatever value it had in the absence of neutrality[10]. On the contrary, as predicted, Majority moves the *fdc* of the problem slightly towards zero, suggesting that OneMax might get harder with this encoding. The question now is: will actual search performance correlate with the *fdc*?

[10]This is not unexpected, since, as discussed in Section V, 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.

Table VI shows the average number of generations required to reach the optimum of OneMax and the percentage of successes in finding the optimum in 1,000 independent runs of an EA. Analysing the results, it can be seen that, when $p_m = 0.01$, there is 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 OneMax problem was solved and average number of generations required to solve it. Also, as predicted by our *fdc* analysis in Section V, the problem is easy and remains easy under all encodings, being solved in almost 100% of cases in all configurations. In addition, it can be seen that, under Majority, more generations are required to solve the problem than under Parity and Truth Table. This, 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. This is another case where *fdc* alone is insufficient.

When $p_m = 0.06$, the situation becomes less clear. Here, the Parity and Truth Table encodings do not perform identically any more, Truth Table still being able to solve the problem in almost all runs, while Parity doing so only in between 70% and 90% of the cases. This, too, 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. Therefore, 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 the number of bits, n, when using the Parity encoding.

These rather confusing trends continue also at the highest genotypic mutation rate, $p_m = 0.1$. 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 an EA, it does not capture all.

To explain these results one really needs to look at phenotypic mutation rates discussed in Section VI. If these are very low, we should expect that mutation will not generate a new phenotype every time it is applied. If new individuals are generated only rarely then evolution will be dominated by selection and the algorithm is likely to converge to a suboptimal solution. If phenotypic mutation rates are very high, the search tends to become almost random. In a unimodal landscape, such as the one associated with OneMax, where the fitness function provides good guidance towards the global optimum, this randomness is very likely to be deleterious. Clearly, the ideal phenotypic mutation rate is somewhere in between these two extremes. In the case of our specific instance of OneMax, the optimal phenotypic mutation rates are perhaps between 0.04 and 0.12.

As one can see in Table I, when $p_m = 0.01$, our bitwise neutrality induces phenotypic mutation rates between 0.0168 and 0.0746. At these mutation rates the EA solves the problem in almost all runs, although the phenotypic mutation rates associated with the Majority encoding (and those in the absence of neutrality) are marginally outside the optimal range and, so, runs last on average slightly longer.

TABLE VII

PERFORMANCE OF AN EA ON THE MULTIMODAL FUNCTION. PAIRS OF NUMBERS IN **boldface**, $\underline{\text{UNDERLINED}}$, DOUBLY $\underline{\underline{\text{UNDERLINED}}}$ AND $\utilde{\text{UNDERLINED}}$ WITH A WAVY LINE REPRESENT SITUATIONS WITH ALMOST IDENTICAL PHENOTYPIC MUTATION RATES.

Type of	$p_m = 0.01$		$p_m = 0.06$		$p_m = 0.1$	
redundancy	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	$\underline{\underline{5.8}}$	44.07	$\utilde{14.2}$
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	$\utilde{12.7}$	46.94	23.2
Truth Table ($n = 5$)	6.41	3.6	15.86	2.5	34.11	$\underline{3.5}$
Truth Table ($n = 6$)	8.18	2.5	20.27	2.2	34.32	$\underline{\underline{4.8}}$
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	$\underline{3.2}$	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

When the genotypic mutation rate is increased to 0.06, the phenotypic mutation rates for the no-neutrality case and for the Majority encoding (see Table I) are still within the optimal range and, thus, performance remains very good. The Truth Table encoding provides phenotypic mutation rates which are marginally outside the optimal range. Here, while success rate remains high, we can see that the number of generations required to find the optimum increases with n. For the Parity encoding, phenotypic mutation rates are way outside the optimal range. So, performance worsens even more, with the higher values of n showing a particularly significant drop in performance. It is then not surprising to see that the EA performs better with Majority (or without neutrality) than with all other encodings.

When $p_m = 0.1$, the phenotypic mutation rates for all encodings are further increased, leading to an even more undirected search. Note, however, that for the no-neutrality case the mutation rate is still within the optimum range and that the phenotypic mutation rates for Truth Table are not too far from it. Thus, performance is still good for these representations. The phenotypic mutation rates associated with Parity are much higher, ranging from 0.3362 for $n = 5$ to 0.4161 in the case $n = 8$. In these conditions the search is almost random and, so, performance is significantly affected by this neutral encoding.

2) The Multimodal Problem Generator: Table VII shows the results of bitwise neutrality on the multimodal problem. Again, at the lowest mutation rate, the predictions of the *fdc* (see Table V (third column)) are roughly correct: the problem is hard ($fdc > 0$) and remains hard irrespective of the encoding used. Also, Parity and Truth Table lead to the same level of difficulty. Again, however, at the higher genotypic mutations rates the situation becomes rather more confusing, with Parity showing improved performance over the other encodings. Furthermore, there is a dependency of performance on n. Effectively, we can observe the opposite effects as in the OneMax problem.

The confusion, again, disappears by considering the phenotypic mutation rates (see Table I) corresponding to

TABLE VIII

PERFORMANCE OF AN EA ON THE TRAP FUNCTION. PAIRS OF NUMBERS IN **boldface**, $\underline{\text{UNDERLINED}}$, DOUBLY $\underline{\underline{\text{UNDERLINED}}}$ AND $\utilde{\text{UNDERLINED}}$ WITH A WAVY LINE REPRESENT SITUATIONS WITH ALMOST IDENTICAL PHENOTYPIC MUTATION RATES.

Type of	$p_m = 0.01$		$p_m = 0.06$		$p_m = 0.1$	
redundancy	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	$\underline{\underline{10.4}}$	44.85	$\utilde{22.0}$
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	$\utilde{20.2}$	46.69	31.5
Truth Table ($n = 5$)	1	0.7	13.05	1.4	41.49	$\underline{6.3}$
Truth Table ($n = 6$)	1.25	0.6	35.16	2.1	47.19	$\underline{\underline{7.8}}$
Truth Table ($n = 7$)	1	0.1	32.36	3.5	47.32	10.9
Truth Table ($n = 8$)	1	0.9	34.44	$\underline{4.8}$	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

each encoding. If these are too low, as for OneMax, evolution will be dominated by selection, resulting in poor performance. Here, however, if phenotypic mutation rates are high, i.e., there is significant randomness in the search, we can expect to escape more easily from local and deceptive optima thereby increasing the success probability. This explains why all encodings are hardly able to solve the problem at $p_m = 0.01$, while some can solve the problem at least some of the times at higher mutation rates. Furthermore, we can now understand why the Parity encoding, having the highest phenotypic mutation rates, does better than all other representations. Finally, we can understand why for all encodings performance increases with n: the phenotypic mutation rate increases as n increases.

3) The Trap Function: As shown in Table VIII, the behaviour of the EA in the Trap problem is a mirror image of that observed on the OneMax problem and it is similar to the behaviour obtained for the multimodal problem. Again, it can be seen how *fdc* (see Table V (fourth column)) makes reasonably good predictions of relative difficulty under different encodings when $p_m = 0.01$, but that the picture becomes less and less clear as p_m increases. However, again, performance differences can be explained easily by considering phenotypic mutation rates. In this case, because the problem is fully deceptive, the more random the search is, the more likely the global optimum will be found. As a result, performance improves as the phenotypic mutation rate increases.

4) MAX-SAT: For space limitations, we will report here the results obtained on MAX-3-SAT problems with 14 variables, to which the results for 7 and 10 were qualitatively very similar. As mentioned before we used clause-to-variable ratios in the range $\{2, 3, 4, 5, 6\}$.

To gain general insights on our chosen MAX-3-SAT class, for each clause-to-variable ratio, we studied 100 random satisfiable problems. The same bitwise-neutrality encodings as for the other problems were used.

Figure 8 reports the *fdc* on different classes of the MAX-SAT problem in the absence of neutrality and when using bitwise neutrality with 3, 5 and 7 bits. For each level of difficulty and encoding, the *fdc* was computed by averaging the *fdc* values estimated over our 100 random problems. For each problem, we estimated the *fdc* via Monte Carlo sampling of 2^ℓ bit strings. As the figure shows, in all conditions, the higher the clauses-to-variables ratio, the higher the *fdc*. This confirms our expectations (based on copious SAT literature) that, in this problem, difficulty increases with the number of clauses (see Section IV-C4). The figure also indicates that, irrespective of the number of bits used, the Parity and Truth Table encodings leave the *fdc* of a problem unchanged, which confirms our expectations from the theory we developed earlier in the paper. Finally, it is interesting to see that, as predicted, the Majority encoding has an effect on the *fdc* of SAT suggesting that the difficulty of the problem is increased by that encoding.

To see to what extent the predictions obtained via an *fdc* analysis of MAX-SAT were correct, we performed a large number of runs of our EA. For $\ell = 14$, we used populations of 128 individuals with runs lasting 64 generations. This ensured that the number of fitness evaluations was exactly half the cardinality of the search space (for 7 and 10 variables, runs were parametrised with the same objective in mind). Again, we used tournaments of size 2 to avoid premature convergence. The same bitwise-neutrality encodings and genotypic mutation rates as for the other problems were used. For each clause-to-variable ratio and on each of the 100 problems with such a ratio, we did 10 independent runs of our mutation-based EA. Thus, in each condition, success rate figures represent the average over 1,000 independent experiments.

A subset of the experimental results thus obtained is reported in Figure 9.[11] Reassuringly, results show that, in all conditions, the higher the clauses-to-variable ratio, the lower the success rate, thereby confirming the predictions based on the *fdc*. We also see that, within experimental errors, the Majority encoding is always the worst (or on par with the worst) of the three bitwise-neutrality encodings, again confirming the prediction of the *fdc*. In other words, *fdc* does a good job at broadly classifying the difficulty of our MAX-SAT problems.

There are situations, however, where the *fdc* was unable to predict relative performance accurately. For example, the *fdc* values for the Parity and Truth Table encodings in Figure 8 are always the same. However, Figure 9 shows that the Parity encoding was the best (or on par with the best) at finding solution for MAX-3-SAT problems in all cases. Also, while the *fdc* in the absence of neutrality was identical to the *fdc* for bitwise neutrality with Parity or Truth Table, as a matter of fact, the addition of neutrality almost always improves performance. Furthermore, performance improves with the number of genotypic bits per phenotypic bit used, while *fdc* remained either constant (Parity and Truth Table) or increased (for Majority). Finally, as for other problems, we see ample variations of performance associated with changes in genotypic mutation rates, which were not (and could not) be predicted by the *fdc*.

Most of these anomalies can only be explained by considering the effects of the encoding on the phenotypic mutation rates. Looking at the no-neutrality case, we see that performance increases with the mutation rate, suggesting that encodings which present a higher phenotypic mutation rate may provide even better performance. With the exception of the no-neutrality case, phenotypic mutation rates are the lowest for the Majority encoding, and

[11]In the figure, we have excluded the results for the clauses-to-variable ratio of 2, since success rate was 100% is all cases. This indicates that the problem is very easy to solve with our parameter settings for the EA.

Encoding			
no neutrality	3 bits	5 bits	7 bits

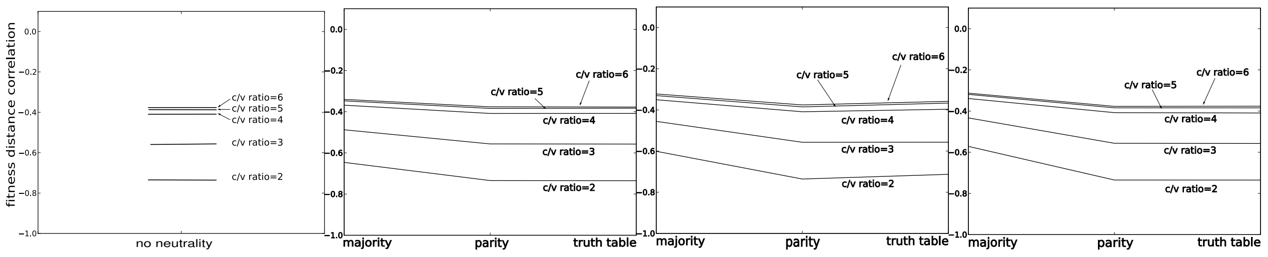


Fig. 8. Fitness distance correlation estimated for MAX-SAT problems of various degrees of difficulty (clause-to-variables ratios) with different encodings.

this is likely to be the reason why such an encoding provides the lowest performance among the bitwise-neutrality representations. Also, the relatively high performance of the Parity encoding is likely to be associated with such an encoding presenting the highest phenotypic mutation rate. The Truth Table encoding positions itself half-way between the other two encodings. Since phenotypic mutation rates grow with the number of bits used, this produces the trends in the bitwise-neutrality plots in Figure 9.

C. Phenotypic Mutation on Rates and Problem Hardness across Problems

In the previous subsections we emphasised the importance of considering the phenotypic mutation rates to complement the information provided by the *fdc*. Figure 10 illustrates how critical this information is.

In particular, Figures 10(a) and (b) plot the success probabilities for the OneMax, Trap and multimodal problems (reported in Tables VI, VII and VIII) against the corresponding genotypic and phenotypic mutation rates, respectively. Distinctions between encodings are totally ignored. Figures 10(c) and (d) report similar results but for MAX-3-SAT problems of different levels of difficulty. It is clear how the success rates for each of the problems studied strongly correlate with the phenotypic mutation rates. Instead, correlation with the genotypic mutation rates is much weaker.

We did not experiment with genotypic mutation rates higher than $p_m = 0.1$. However, it is easy to predict what would happen. The Parity and Truth Table encodings would progressively move towards phenotypic mutation rates of 50%, making the search effectively random. We can easily compute the expected success rate of the EA in these conditions. In the OneMax, Trap and multimodal problems, in each EA run we do 8,000 trials (80 individuals for 100 generations), which represent 48.82% of the search space size, 2^ℓ, since $\ell = 14$. This is an *upper limit* for the success rate. Of course, because of re-sampling, we should only expect to find the optimum with a lower probability. The exact value is 38.3%, as one can compute using the theory for the coupon collector problem (see [61]). So, 38.3% is the limit performance for high mutation rates for these problems. Note that this limit is problem independent as long as a problem has a single global optimum as in the case of the three problems mentioned above. Indeed, the three plots in Figure 10(b) show a convergence towards this limit.

	Encoding			
c/v	no neutrality	3 bits	5 bits	7 bits

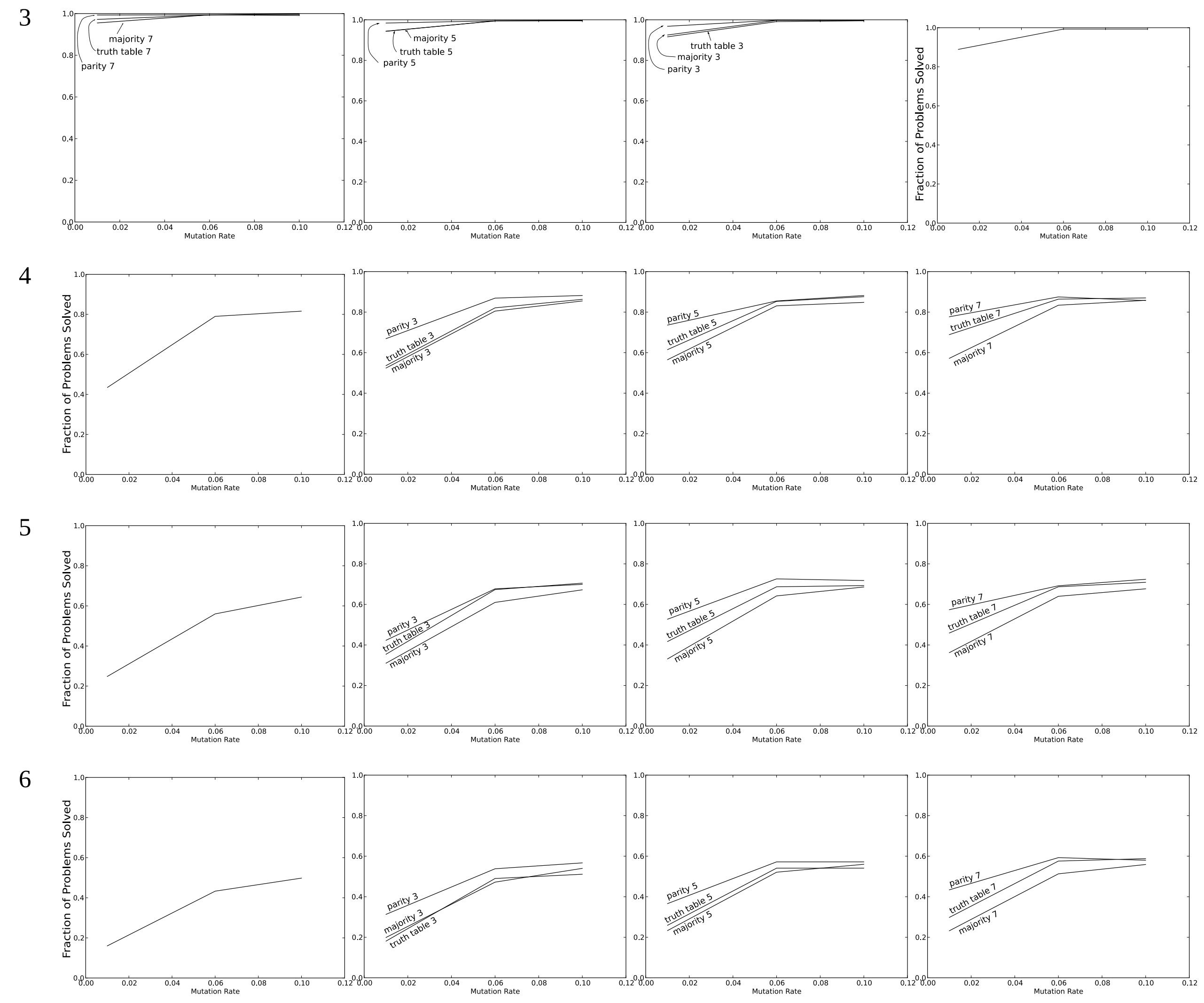
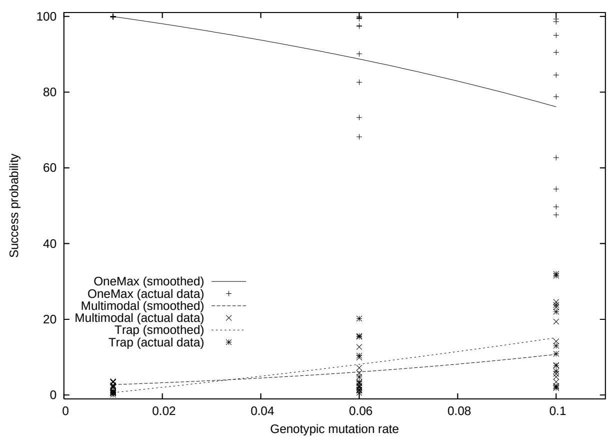
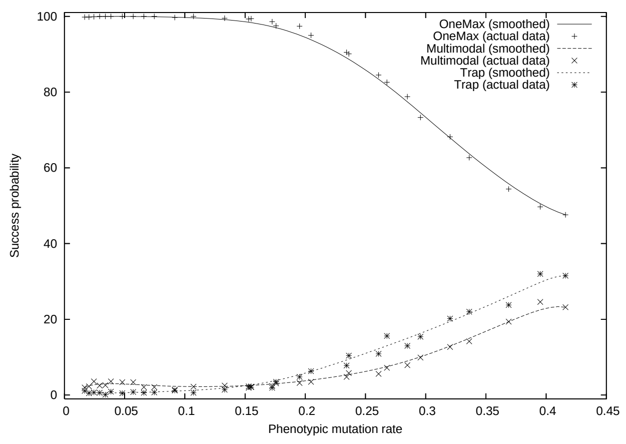


Fig. 9. Success rates for MAX-SAT problems of varying difficulty (clause-to-variables ratios) with different encodings and genotypic mutation rates.

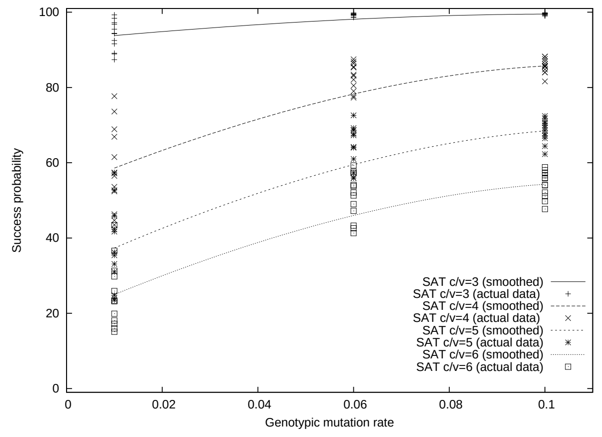
For MAX-3-SAT the situation is very similar, but there is not a unique limit to the success rate as the search becomes more and more random. This is because the limit success rate depends on the average number of global optima (assignments that satisfy a formula) in the search space and the higher the clause-to-variables ratio, the fewer the global optima. So, we find that the limit decreases as the ratio increases and *vice versa*. It is interesting to see in Figure 10(d) that, in most cases, the limit is rapidly approached as the phenotypic mutation rate increases. The plots thus present a knee. Interestingly, this is not too far from $1/\ell \cong 0.0714$, which, on average, corresponds



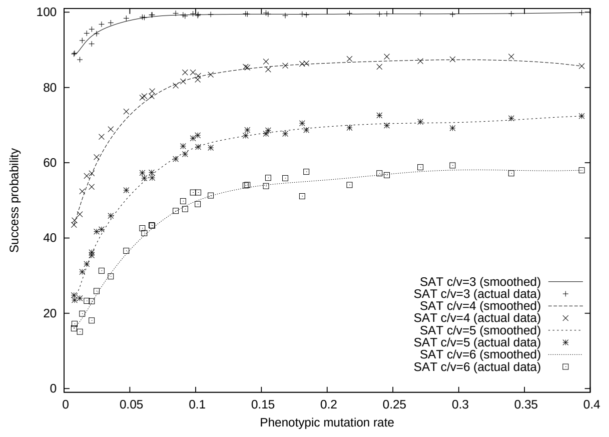
(a)



(b)



(c)



(d)

Fig. 10. Plots of the success probability as a function of the genotypic mutation rate (a) and the phenotypic mutation rate (b) for the OneMax, Multimodal and Trap fitness functions, and corresponding plots for MAX-3-SAT with 14 variables (c–d).

to a single variable flip per mutation. At this mutation level, the EA's search strategy becomes quite similar to the one-flip strategy adopted by many modern local search heuristics for SAT.

VIII. Discussions

In this section we discuss the results reported above from both the point of view of understanding neutrality and the practical consequences of the work in relation to the solution of real-world problems. In particular, we want to highlight what has and has not been achieved with this study from these viewpoints.

To make the problem as simple as possible, we adopted the strategy of using one of the simplest possible EAs and extremely simple encodings to increase the neutrality of a search space. Nonetheless, both our theoretical derivations and our empirical results indicate that changing the problem or performing small changes in the details of the representation and search parameters used can turn a neutral encoding from being beneficial to being disadvantageous and *vice versa*. That the neutrality could not always be helpful is an obvious consequence of the no-free-lunch theory [62], but that the performance of an EA in the presence of neutrality could be so sensitive to details was much less expected. This is probably the key source of the confusion and controversy reigning in the sizable literature on the use of neutrality in EAs.

Our results indicate that, in many conditions, *fdc* provides a rough indication of problem difficulty and of how problem difficulty is affected by neutrality. However, we have also found that, in order to obtain more accurate information, one also needs to consider how the chosen representation translates genotypic mutations into phenotypic ones.

For some representations, *fdc* and phenotypic mutation rates can be computed theoretically, while for others they need to be estimated via sampling. Theoretical formulations are particularly important because they reveal how specific details of the representation (including the value of the fitness of the neutral network in constant neutrality, the details of the encoding function used in bitwise neutrality and the number of genotypic bits used in the encoding) influence the features of the landscape and search operators. Another advantage of theoretical formulations is that their complexity is often scale-invariant, unlike empirical evaluations where computational cost may prevent the analysis of large, realistic problems.

Both theoretical analyses and empirical estimations of *fdc*, however, are basically impossible for real-world problems where, typically, the position of the global optima is unknown. Other predictive performance measures exist which do not require knowledge of the optima (e.g., see [63]–[67]), but, to the best of our knowledge, they have not been used to study the effects of neutrality so far. Also, fitness landscapes can be studied via the analysis of their Walsh coefficients. For some problems, for example SAT [68], this can be done efficiently for large scale instances, although we are not aware of any Walsh analysis relating to neutrality. These are certainly areas that future research on neutrality could beneficially target to gather insights on its benefits and drawbacks in relation to real-world problems.

While *fdc* cannot really be applied to real-world problems, the notion of phenotypic mutation rate can, at least in the case of bitwise neutrality. This is because, in this form of neutrality, phenotypic mutation rates are only a function of the representation adopted, not the fitness function (while in constant neutrality there is a partial dependency on fitness). So, our calculations in Section VI-B are applicable to problems of any size and in any application relying on a binary encoding.

For a mutation-based EA, trivial neutrality (see Section II) is ineffective at changing the difficulty of a problem, provided one ensures the phenotypic mutation rate used with neutrality matches the mutation rate with the original representation. If this is not the case, then behaviour and performance can change very significantly.

Non-trivial neutrality (e.g., Majority), is neither *a priori* beneficial nor disadvantageous. It can be beneficial in problems where the original landscape is misleading/deceptive, but only if it is set up in such a way to ensure that a large enough proportion of the population spends long enough exploring neutral networks. In a mutation-only EA, the exploration of neutral networks is essentially a random-walk type of search. While normally this form of search is inefficient, in the presence of a misleading fitness function, it is the only hope for a population to eventually locate superior areas in the search space (e.g., the global optimum). However, having neutral networks in the search space does not mean that the population will stay on them. This will not happen, for example, if neutral networks have a particularly low fitness. In such cases, it is unlikely that neutrality will benefit a deceptive problem.

In easy problems, where the fitness function provides reliable guidance towards good areas of the search space, it is unlikely that non-trivial neutrality will provide any benefits irrespective of whether neutral networks have high fitness or not. This is for the following reasons. Random search is inefficient for problems where the fitness function provides good guidance. In such problems, if neutral networks have high fitness, a part of the population will randomly drift on them, thus reducing the overall efficiency of the search. If, instead, neutral networks are unfit, selection will avoid giving individuals on them a chance to mutate. Such individuals are, therefore, wasted samples, which reduce the efficiency of the search.

While we have introduced and used the notion of phenotypic mutation rate as an atomic quantity, in fact, the phenotypic mutation rates we computed are averages across all possible strings. This, however, does not mean that all strings have the same probability of being mutated into a string that represents a different phenotype. For example, in Figure 11 we show the distribution of phenotypic mutation rates with 7-bit bitwise neutrality for different encodings and genotypic mutation rates. From this figure it is immediate to see that, except for the Parity encoding, some strings may be exceedingly robust to genotypic mutations (in the sense that they are more likely to represent the same phenotype after mutation) while others are extremely fragile. For example, under the Majority encoding the string 00000 can have any bit flipped and still remain in the same majority class while 3 out of 5 times flipping one bit in the string 00011 will change its phenotypic class.

To some extent, we would expect evolution to exploit these differences. For example, it could protect certain particular genes or highly epistatic groups of genes by using a very robust genotypic representation for them, while keeping the phenotypic mutation rate high for suboptimal phenotypic bits by using fragile encodings. However, at this stage, we do not know how important this effect is in a mutation-based EA. Also, while it is likely that the addition of recombination will further emphasise the importance of representational inhomogeneities, we have not turned this particular stone.

This leads to another unturned stone in our work. Following from our earlier work, recently [69] have highlighted the possibility and benefits of using different numbers of bits for different parts of a representation with bitwise neutrality.[12] It would be interesting to see what effects this has on the *fdc* of a problem. Also, while [69] considered the adoption of different numbers of bits as roughly equivalent to providing different parts of the representation with different mutation rates, as shown in Figure 11, the equivalence is only correct an average. It would be interesting to see if the results of [69] could be extended to consider phenotypic mutation rate distributions.

In some sense all this is related to the point of view developed in [17] who 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 choose in a self-adaptive way. In the future, it would be interesting to reinterpret our results by looking at neutrality as adaptation of the representation.

[12] They have also reported a closed form representation for one of our results on phenotypic mutation rates and inverted our formulae.

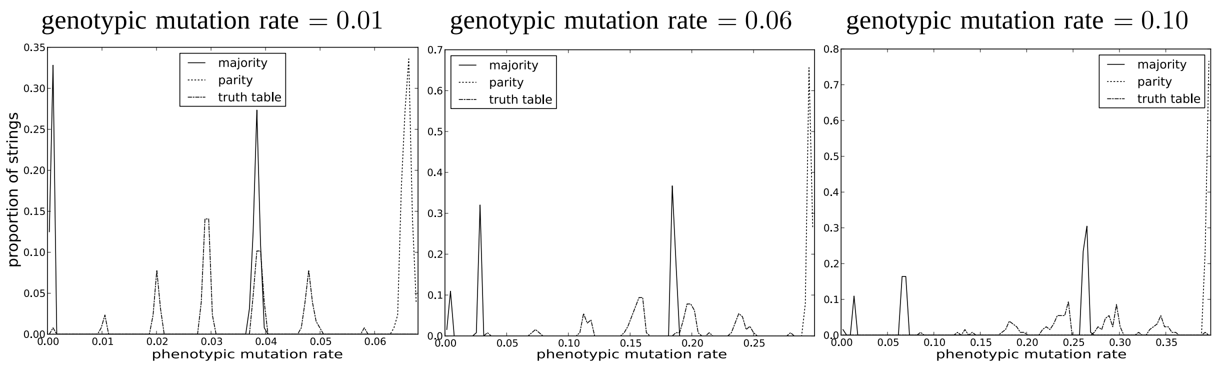


Fig. 11. Distribution of phenotypic mutation rates with 7-bit bitwise neutrality for different encodings and genotypic mutation rates.

IX. Conclusions

Does neutrality help or hinder the search of an EA? This question has been debated at considerable length in the literature without really reaching any form of consensus on its answer.

As we discussed in Sections I and II, the reasons for this situation include the lack of a single definition of neutrality, the multiple ways in which one can add neutrality to a representation, the focus on pure performance when evaluating the effects of neutrality without attention to the changes in the behaviour of the search operators and in the features of the fitness landscape, and, finally, the variability in the choice of problems, algorithms and representations for benchmarking purposes. Also, very often studies consider problems and representations that are quite complex and results represent the composition of multiple effects.

In this paper, we have attempted to address these problems and to shed some light on neutrality. Our strategy has been as follows. Firstly, we have used one of the simplest evolutionary algorithms — a mutation-only binary EA. Secondly, we have considered only extremely simple representations that can be used to artificially increase the neutrality in a system: constant neutrality, where neutrality is plugged into the original encoding by adding an extra bit to the representation, and bitwise neutrality, where each phenotypic bit is obtained by transforming a group of genotypic bits via an encoding function. Thirdly, we have studied neutrality both theoretically — via fitness-distance correlations and phenotypic mutation rates — and empirically (using both standard benchmarks for binary EAs and the class of MAX-SAT problems, which is more interesting for practitioners), and made an effort to integrate the results of the two viewpoints.

The key lessons we have learnt during our explorations are the following. Firstly, we have found that *fdc* is often able to predict the relative difficulty of problems correctly with and without the addition of neutrality. For example, in the case of MAX-SAT instances, it correctly ranked them by difficulty in total agreement with the well-known clause-to-variables ratio criterion. Secondly, we found that *fdc* was not able to predict the fine details of the behaviour of our mutation-based EA, unavoidably neglecting the influence that mutation rates have on performance. Thirdly, we understood that by complementing the *fdc* analysis with an analysis of the changes in phenotypic mutation rates associated with representation changes provides a much clearer picture of the effects of the two forms of neutrality we studied. Fourthly, we have found that the performance of an EA in the presence of neutrality is extremely sensitive to details such as the problem being solved, the type of encoding adopted and the rate of application of operators. Therefore, the question of whether neutrality helps or hinders the search in EAs is ill-posed and cannot be answered in general: one can only answer this question within the context of a specific class of problems, (neutral) representation and set of operators.

While we feel that these lessons are quite useful and general, many avenues have been left unexplored by this study. The effects of recombination, for example, have not been analysed. Also, the effects of the distribution of phenotypic mutation rates have not been considered in detail. We have made an effort to include in the analysis also a real-world application domain (MAX-SAT), but the size of the instances we considered had to be very small to make the *fdc* analysis viable. Furthermore, we have only studied two forms of neutrality, while many other interesting forms exist, including, for example, the encoding of search parameters (such as mutation rates) in the chromosomes which was studied in [17]. Finally, we have not looked at the effects of neutrality in noisy or dynamic fitness functions. All of these areas would be worthy of future research.

References

[1] M. Kimura, "Evolutionary rate at the molecular level," in *Nature*, vol. 217, 1968, pp. 624–626.

[2] ——, *The Neutral Theory of Molecular Evolution*. Cambridge, UK: Cambridge University Press, 1983.

[3] T. Yu and J. F. Miller, "Finding Needles in Haystacks is not Hard with Neutrality," in *Genetic Programming Proceedings of the 5th European Conference, EuroGP 2002*, ser. LNCS, J. A. Foster, E. Lutton, J. F. Miller, C. Ryan, and A. Tettamanzi, Eds., vol. 2278. Kinsale, Ireland: Springer-Verlag, 3-5 Apr. 2002, pp. 13–25.

[4] J. F. Miller, "An Empirical Study of the Efficiency of Learning Boolean Functions Using a Cartesian Genetic Approach," in *Proceedings of the Genetic and Evolutionary Computation Conference GECCO'99*, W. Banzhaf, J. M. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. J. Jakiela, and R. E. Smith, Eds., vol. 2. Orlando, Florida: Morgan Kaufmann, 13-17 Jul. 1999, pp. 1135–1142.

[5] J. F. Miller and P. Thomson, "Cartesian genetic programming," in *Third European Conference on Genetic Programming EuroGP 2000*, ser. LNCS, R. Poli, W. Banzhaf, W. Langdon, J. Miller, P. Nordin, and T. Fogarty, Eds., vol. 1802. Edinburgh: Springer-Verlag, 15-16 Apr. 2000, pp. 121–132.

[6] M. Collins, "Finding Needles in Haystacks is Harder with Neutrality," in *GECCO 2005: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, H.-G. Beyer, U.-M. O'Reilly, D. V. Arnold, W. Banzhaf, C. Blum, E. W. Bonabeau, E. Cantu-Paz, D. Dasgupta, K. Deb, J. A. Foster, E. D. de Jong, H. Lipson, X. Llora, S. Mancoridis, M. Pelikan, G. R. Raidl, T. Soule, A. M. Tyrrell, J.-P. Watson, and E. Zitzler, Eds., vol. 2. Washington DC, USA: ACM Press, 25-29 Jun. 2005, pp. 1613–1618.

[7] W. B. Langdon and R. Poli, *Foundations of Genetic Programming*. Berlin: Springer, 2002.

[8] R. Poli, W. B. Langdon, and N. F. McPhee, *A Field Guide to Genetic Programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008, (With contributions by J. R. Koza).

[9] 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*, M. Keijzer, M. Cattolico, D. Arnold, V. Babovic, C. Blum, P. Bosman, M. V. Butz, C. C. Coello, D. Dasgupta, S. G. Ficici, J. Foster, A. Hernandez-Aguirre, G. Hornby, H. Lipson, P. McMinn, J. Moore, G. Raidl, F. Rothlauf, C. Ryan, and D. Thierens, Eds. Seattle, WA, USA: ACM Press, 8-12 Jul. 2006, pp. 1149–1156.

[10] ——, "Some Steps Towards Understanding How Neutrality Affects Evolutionary Search," in *Parallel Problem Solving from Nature (PPSN IX). 9th International Conference*, ser. LNCS, T. P. Runarsson, H.-G. Beyer, E. Burke, J. J. Merelo-Guervós, L. D. Whitley, and X. Yao, Eds., vol. 4193. Reykjavik, Iceland: Springer-Verlag, 9-13 Sep. 2006, pp. 778–787.

[11] R. Poli and E. Galván-López, "On The Effects of Bit-Wise Neutrality on Fitness Distance Correlation, Phenotypic Mutation Rates and Problem Hardness," in *Foundations of Genetic Algorithms IX*, ser. Lecture Notes in Computer Science, C. R. Stephens, M. Toussaint, D. Whitley, and P. Stadler, Eds. Mexico city, Mexico: Springer-Verlag, 8-11 Jan. 2007, pp. 138–164.

[12] E. Galván-López, "An analysis of the effects of neutrality on problem hardness for evolutionary algorithms," Ph.D. dissertation, School of Computer Science and Electronic Engineering, University of Essex, United Kindgom, 2009.

[13] E. V. Nimwegen, J. P. Crutchfield, and M. Huynen, "Neutral Evolution of Mutational Robustness," *Proc. Natl. Acad. Sci. USA*, vol. 96, no. 17, pp. 9716–9720, 1999.

[14] A. Wagner, "Robustness, Evolvability and Neutrality," *FEBS Letters*, vol. 579, no. 8, pp. 1772–1778, 2005.

[15] C. Reidys, P. F. Stadler, and P. Schuster, "Generic Properties of Combinatory Maps – Neutral Networks of RNA Secondary Structures," *Bull. Math. Biol.*, vol. 59, pp. 339–397, 1997.

[16] P. Schuster, W. Fontana, P. F. Stadler, and I. L. Hofacker, "From Sequences to Shapes and Back: A Case Study in RNA Secondary Structures," *Royal Society of London Proceedings Series B*, vol. 255, pp. 279–284, Mar. 1994.

[17] M. Toussaint and C. Igel, "Neutrality: A necessity for self-adaptation," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002)*, 2002, pp. 1354–1359.

[18] A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms." *IEEE Trans. Evolutionary Computation*, vol. 3, no. 2, pp. 124–141, 1999.

[19] H.-G. Beyer, *The Theory of Evolution Strategies*. Springer Verlag, 2001.

[20] M. Toussaint, "On the Evolution of Phenotypic Exploration Distributions," in *Foundations of Genetic Algorithms 7 (FOGA 2003)*, K. A. De Jong, R. Poli, and J. Rowe, Eds. Morgan Kaufmann, 2003, pp. 169–182.

[21] M. D. Vose, *The simple genetic algorithm: Foundations and theory*. Cambridge, MA: MIT Press, 1999.

[22] M. A. Shackleton, R. Shipman, and M. Ebner, "An Investigation of Redundant Genotype-Phenotype Mappings and Their Role in Evolutionary Search," in *Proceedings of the International Congress on Evolutionary Computation (CEC 2000)*, A. Zalzala, C. Fonseca, J. H. Kim, and A. Smith, Eds. IEEE Press, 2000, pp. 493–500.

[23] R. Shipman, M. Shackleton, M. Ebner, and R. Watson, "Neutral Search Spaces for Artificial Evolution: A Lesson From Life," in *Artificial Life: Proceedings of the Seventh International Conference on Artificial Life*, M. Bedau, S. Rasmussen, J. McCaskill, and N. Packard, Eds. MIT Press, 2000, pp. 162–169.

[24] M. Ebner, P. Langguth, J. Albert, M. Shackleton, and R. Shipman, "On Neutral Networks and Evolvability," in *Proceedings of the 2001 IEEE Congress on Evolutionary Computation*. IEEE Press, 27-30May 2001, pp. 1–8.

[25] M. Ebner, M. Shackleton, and R. Shipman, "How Neutral Networks Influence Evolvability," *Complexity*, vol. 7, no. 2, pp. 19–33, 2001.

[26] J. D. Knowles and R. A. Watson, "On the Utility of Redundant Encodings in Mutation-Based Evolutionary Search," in *Parallel Problem Solving from Nature - PPSN VII: 7th International Conference*, J. J. M. Guervós, P. Adamidis, H.-G. Beyer, J. L. F.-V. Martín, and H.-P. Schwefel, Eds. Granada, Spain: Springer Verlag, 2002, pp. 88–98.

[27] F. Rothlauf and D. Goldberg, "Redundant Representations in Evolutionary Algorithms," *Evolutionary Computation*, vol. 11, no. 4, pp. 381–415, 2003.

[28] C. Fonseca and M. Correia, "Developing Redudant Binary Representations for Genetic Search," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*. Edinburgh: IEEE, 2-4 Sep. 2005, pp. 372–379.

[29] R. Shipman, "Genetic Redundancy: Desirable or Problematic for Evolutionary Adaptation," in *4th International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA'99)*, A. Dobnikar, N. C. Steele, D. W. Pearson, and R. F. Albrecht, Eds. Springer-Verlag, 1999, pp. 337–344.

[30] E. Galván-López, R. Poli, A. Kattan, M. O'Neill, and A. Brabazon, "Neutrality in evolutionary algorithms ... what do we know?" *Evolving Systems*, 2011. (accepted).

[31] T. Jones, "Evolutionary algorithms, fitness landscapes and search," Ph.D. dissertation, University of New Mexico, Albuquerque, 1995.

[32] 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*, L. J. Eshelman, Ed. San Francisco, CA, USA: Morgan Kaufmann Publishers, 1995, pp. 184–192.

[33] P. Collard, A. Gaspar, M. Clergue, and C. Escazut, "Fitness Distance Correlation as Statistical Measure of Genetic Algorithms Difficulty, Revisited," in *Proceedings of the European Conference on Artificial Intelligence*. John Witley & Sons, Ltd, 1998, pp. 650–654.

[34] Clergue and Collard, "Genetic Heuristic for Search Space Exploration," in *Proceedings of International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 1999, pp. 1218–1224.

[35] M. Finger, T. Stutzle, and H. Lourenco, "Exploiting Fitness Distance Correlation of Set Covering Problems," in *Proceedings of the Applications of Evolutionary Computing on EvoWorkshops 2002*. London, UK: Springer-Verlag, 2002, pp. 61–71.

[36] W. Beaudoin, S. Verel, P. Collard, and C. Escazut, "Deceptiveness and Neutrality The ND Family of Fitness Landscapes," in *GECCO 2006: Proceedings of the 2006 Conference on Genetic and Evolutionary Computation*, M. Keijzer, M. Cattolico, D. Arnold, V. Babovic, C. Blum, P. Bosman, M. V. Butz, C. C. Coello, D. Dasgupta, S. G. Ficici, J. Foster, A. Hernandez-Aguirre, G. Hornby, H. Lipson, P. McMinn, J. Moore, G. Raidl, F. Rothlauf, C. Ryan, and D. Thierens, Eds., vol. 1. Seattle, WA, USA: ACM Press, 8-12 Jul. 2006, pp. 507–514.

[37] M. Clergue, P. Collard, M. Tomassini, and L. Vanneschi, "Fitness Distance Correlation and Problem Difficulty for Genetic Programming," in *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2002*, W. B. Langdon, E. Cantú-Paz, K. E. 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. K. Burke, and N. Jonoska, Eds. New York: Morgan Kaufmann Publishers, 9-13 Jul. 2002, pp. 724–732.

[38] L. Vanneschi and M. Tomassini, "Pros and Cons of Fitness Distance Correlation in Genetic Programming," in *2003 Genetic and Evolutionary Computation Conferencem Workshop Program Proceedings, GECCO'03*, A. M. Barry, Ed. Chigaco: AAAI, 11 Jul. 2003, pp. 284–287.

[39] L. Vanneschi, M. Tomassini, M. Clergue, and P. Collard, "Difficulty of Unimodal and Multimodal Landscapes in Genetic Programming," in *Genetic and Evolutionary Computation – GECCO-2003*, ser. LNCS, E. Cantú-Paz, J. A. Foster, K. Deb, D. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. Dowsland, N. Jonoska, and J. Miller, Eds., vol. 2724. Chicago: Springer-Verlag, 12-16 Jul. 2003, pp. 1788–1799.

[40] L. Vanneschi, M. Tomassini, P. Collard, and M. Clergue, "Fitness Distance Correlation in Structural Mutation Genetic Programming," in *Proceedings of the sixth European Conference on Genetic Programming, EuroGP 2003*, ser. LNCS, C. Ryan, T. Soule, M. Keijzer, E. P. K. Tsang, R. Poli, and E. Costa, Eds., vol. 2610. Essex: Springer-Verlag, 14-16 Apr. 2003, pp. 455–464.

[41] L. Vanneschi, "Theory and practice for efficient genetic programming," Ph.D. dissertation, Faculty of Science, University of Lausanne, Switzerland, 2004.

[42] M. Tomassini, L. Vanneschi, P. Collard, and M. Clergue, "A Study of Fitness Distance Correlation as a Difficulty Measure in Genetic Programming," *Evolutionary Computation*, vol. 13, no. 2, pp. 213–239, Summer 2005.

[43] E. Galván-López, S. Dignum, and R. Poli, "The Effects of Constant Neutrality on Performance and Problem Hardness in GP," in *EuroGP 2008 - 11th European Conference on Genetic Programming*, ser. LNCS, M. ONeill, L. Vanneschi, S. Gustafson, A. I. E. Alcazar, I. D. Falco, A. D. Cioppa, and E. Tarantino, Eds., vol. 4971. Napoli, Italy: Springer, 26–28 Mar. 2008, pp. 312–324.

[44] L. Altenberg, "Fitness Distance Correlation Analysis: An Instructive Counterexample," in *Proceedings of the Seventh International Conference on Genetic Algorithms*, T. Back, Ed. San Francisco, CA, USA: Morgan Kaufmann, 1997, pp. 57–64.

[45] R. J. Quick, V. J. Rayward-Smith, and G. D. Smith, "Fitness Distance Correlation and Ridge Functions," in *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*. London, UK: Springer-Verlag, 1998, pp. 77–86.

[46] 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*, M. Keijzer, M. Cattolico, D. Arnold, V. Babovic, C. Blum, P. Bosman, M. V. Butz, C. C. Coello, D. Dasgupta, S. G. Ficici, J. Foster, A. Hernandez-Aguirre, G. Hornby, H. Lipson, P. McMinn, J. Moore, G. Raidl, F. Rothlauf, C. Ryan, and D. Thierens, Eds. Seattle, WA, USA: ACM Press, 8-12 Jul. 2006, pp. 1233–1240.

[47] K. A. De Jong, M. A. Potter, and W. M. Spears, "Using Problem Generators to Explore the Effects of Epistasis," in *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97)*, T. Bäck, Ed. San Francisco, USA: Morgan Kaufmann, 1997, pp. 338–345.

[48] 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*. Piscataway, NJ: IEEE Press, 1998, pp. 78–83.

[49] D. E. Goldberg, "Construction of High-Order Deceptive Functions Using Low-Order Walsh Coefficients." *Ann. Math. Artif. Intell.*, vol. 5, no. 1, pp. 35–47, 1992.

[50] D. E. Goldberg, K. Deb, and J. Horn, "Massive Multimodality, Deception, and Genetic Algorithms," in *PPSN II: Proceedings of the 2nd International Conference on Parallel Problem Solving from Nature*, R. Männer and B. Manderick, Eds. Amsterdam: Elsevier Science Publishers, B. V., 1992, pp. 37–48.

[51] H. Kargupta, K. Deb, and D. Goldberg, "Ordering Genetic Algorithms and Deception," in *PPSN II: Proceedings of the 2nd International*

Conference on Parallel Problem Solving from Nature, R. Männer and B. Manderick, Eds. Amsterdam: Elsevier Science Publishers, B. V., 1992, pp. 49–58.

[52] M. Davis, G. Logemann, and D. Loveland, “A machine program for theorem-proving,” *Commun. ACM*, vol. 5, no. 7, pp. 394–397, 1962.

[53] S. A. Cook, “The complexity of theorem-proving procedures,” in *STOC ’71: Proceedings of the third annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 1971, pp. 151–158.

[54] B. Selman, H. Levesque, and D. Mitchell, “A new method for solving hard satisfiability problems,” in *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI’92)*, 1992, pp. 459–465.

[55] B. Selman, H. A. Kautz, and B. Cohen, “Noise strategies for improving local search,” in *AAAI ’94: Proceedings of the twelfth national conference on Artificial intelligence (vol. 1)*. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1994, pp. 337–343.

[56] H. H. Hoos and T. Stützle, “Local search algorithms for SAT: An empirical evaluation,” *Journal of Automated Reasoning*, vol. 24, no. 4, pp. 421–481, 2000.

[57] J. Gottlieb, E. Marchiori, and C. Rossi, “Evolutionary algorithms for the satisfiability problem,” *Evol. Comput.*, vol. 10, no. 1, pp. 35–50, 2002.

[58] M. B. Bader-El-Den and R. Poli, “Generating sat local-search heuristics using a gp hyper-heuristic framework,” in *Artificial Evolution*, ser. Lecture Notes in Computer Science, N. Monmarché, E.-G. Talbi, P. Collet, M. Schoenauer, and E. Lutton, Eds., vol. 4926. Springer, 2007, pp. 37–49.

[59] D. G. Mitchell, B. Selman, and H. J. Levesque, “Hard and easy distributions for SAT problems,” in *Proceedings of the Tenth National Conference on Artificial Intelligence*, P. Rosenbloom and P. Szolovits, Eds., American Association for Artificial Intelligence. Menlo Park, California: AAAI Press, 1992, pp. 459–465.

[60] B. Selman, H. Kautz, and B. Cohen, “Local search strategies for satisfiability testing,” in *AAAI-92: Proceedings 10th National Conference on AI*, ser. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, Jan. 1995.

[61] W. Feller, *An Introduction to Probability Theory and Its Applications*. Wiley, 1968.

[62] D. Wolpert and W. Macready, “No free lunch theorems for optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, April 1997.

[63] E. Galván-Lopéz, J. McDermott, M. O’Neill, and A. Brabazon, “Defining locality in genetic programming to predict performance,” in *CEC 2010: Proceedings of the 12th Annual Congress on Evolutionary Computation*. Barcelona, Spain: IEEC Press, July 2010.

[64] ——, “Towards an understanding of locality in genetic programming,” in *GECCO 2010: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*. Portland, Oregon, USA: ACM Press, July 2010.

[65] E. Galván-López, J. McDermott, M. O’Neill, and A. Brabazon, “Defining locality in problem hardness in genetic programming,” *Genetic Programming and Evolvable Machines*, 2011. (accepted).

[66] R. Poli and L. Vanneschi, “Fitness-proportional negative slope coefficient as a hardness measure for genetic algorithms,” in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2007)*, H. Lipson, Ed. ACM, 2007, pp. 1335–1342.

[67] L. Vanneschi, M. Tomassini, P. Collard, S. Verel, Y. Pirola, and G. Mauri, “A comprehensive view of fitness landscapes with neutrality and fitness clouds,” in *Proceedings of EuroGP 2007*, ser. LNCS, vol. 4445. Springer, 2007, pp. 241–250.

[68] S. Rana, R. B. Heckendorn, and D. Whitley, “A tractable walsh analysis of SAT and its implications for genetic algorithms,” in *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI’98)*, 1998, pp. 392–397.

[69] T. Friedrich and F. Neumann, “When to use bit-wise neutrality,” *Natural Computing*, vol. 9, no. 1, pp. 283–294, 2010.

List of Figures