Fitness-Proportional Negative Slope Coefficient as a Hardness Measure for Genetic Algorithms

Riccardo Poli
Department of Computer Science
University of Essex, UK
rpoli@essex.ac.uk

Leonardo Vanneschi
Dipartimento di Informatica, Sistemistica e Comunicazione (D. I. S. Co.)
University of Milano-Bicocca, Milan, Italy
vanneschi@disco.unimib.it

ABSTRACT

The Negative Slope Coefficient (*nsc*) is an empirical measure of problem hardness based on the analysis of offspring-fitness vs. parent-fitness scatterplots. The *nsc* has been tested empirically on a large variety problems showing considerable reliability in distinguishing easy from hard problems. However, neither a theoretical justification nor a theoretical analysis of the *nsc* have ever been given. This paper presents a modification of *nsc*, the fitness-proportional negative slope coefficient (*fpncs*), for which it is possible to give a theoretical explanation and analysis. To illustrate the approach we compute *fpnsc* theoretically for the class of invertible functions of unitation, and for two mutation operators. We apply the theory to compute *fpnsc* for three benchmark functions: Onemax, Trap and Onemix. We then compare the predictions of *fpnsc* with the success probability recorded in actual runs. The results suggest that *fpnsc* is able to broadly discriminate between easy and hard GA problems.

Categories and Subject Descriptors

I.2.m.c [**Artificial Intelligence**]: Miscellaneous: evolutionary computing and genetic algorithms

General Terms

Algorithms, Performance

Keywords

Negative slope coefficient, theory, problem difficulty

1. INTRODUCTION

For classical algorithms a well-developed theory exists to categorise problems into complexity classes. Problems in the same class have roughly the same complexity, i.e., they consume (asymptotically) the same amount of computational resources, usually time [15]. Although, properly speaking, Evolutionary Algorithms (EAs) are randomised heuristics and not algorithms, it would be useful to be somehow able to likewise classify problems according to some measure of difficulty. Difficulty studies in Genetic Algorithms (GAs) have been pioneered by Goldberg and coworkers (e.g., see [4, 11]). Their approach consisted in constructing functions that should a priori be easy or hard for GAs to solve. These ideas have been followed by many others (e.g. [14, 7]) and have been at least partly successful in the sense that they have been the source of a considerable amount of other work on what makes a problem easy or difficult for GAs. One concept that underlies many approaches is the notion of *fitness landscape*, originally proposed in [28].

The fitness landscape metaphor can be helpful to understand the difficulty of a problem for a searcher that is trying to find the optimal solution for that problem. For example, imagine a very smooth and regular landscape with a single hill top. This is the typical fitness landscape of an easy problem: most search strategies (hill climbing, simulated annealing, EAs, etc.) are able to find the top of the hill in a straightforward manner. The opposite is true for a very rugged landscape, with many hills and local optima which are not as high as the best one. In this case, even approaches based on populations of individuals, like GAs or GP, might have problems.

The graphical visualisation of fitness landscapes, whenever possible, can give an indication about the difficulty of a problem for a searching agent like EAs. However, even assuming that one is able to draw a fitness landscape (which is generally not the case, given the huge sizes of typical search spaces and neighbourhoods), the mere observation of its graph surely lacks formality [9, 6]. The ideal situation would be to have a numeric measure able to condense useful information on fitness landscapes.

In [12], Jones introduced a heuristic called *fitness distance correlation* (*fdc*), as an algebraic indicator of problem difficulty for GAs. Those studies have been extended to GP, for instance, in [20, 21]. *fdc* can be considered quite a reliable indicator of problem hardness. However, it has some flaws, the most severe one being that the optimal solution (or solutions) must be known beforehand. This is obviously unrealistic in applied search and optimisation problems, and prevents one from applying *fdc* to more usual benchmarks and real-life applications. Thus, although the study of *fdc* is useful, it is also important to try other approaches based on quantities that can be measured without any explicit knowledge of the genotype of optimal solutions.

One measure that does not require knowledge of landscape optima, the *negative slope coefficient* (*nsc*), was introduced

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

in [22, 23, 21]. It is closely related to the notion of evolvability and it has given some promising results on a variety of GP problems. However, this measure has neither theoretically been justified nor assessed on genetic algorithms.

The origin, a definition and a brief discussion of *nsc* and of a slightly modified version of it, that we call *fitness proportional nsc* (*fpnsc*) is given in Section 2. In Section 3, we model *fpnsc* theoretically and provide a simple explanatory example. Then (Section 4) we use the theory to compute *fpnsc* for all invertible functions of unitation, when the genetic operator used is a single, random, bit-flip. We compute *fpnsc* for the same class of functions but for standard bit-flip mutation in Section 5. In both cases we compare the predicted difficulty with empirical measures of the actual success rates of a GA using these operators. We discuss the benefits and drawbacks of *fpnsc* in Section 6. We draw some conclusions in Section 7.

2. NSC AND FPNSC

Evolvability is a feature that is intuitively related, although not exactly identical, to problem difficulty. It has been defined as the ability of genetic operators to improve fitness quality [1]. The most natural way to study evolvability is, probably, to plot the fitness values of individuals against the fitness values of their neighbours, where a neighbour is obtained by applying one step of a genetic operator to the individual. Such a plot has been presented in [25, 3, 24, 2] and it is called a *fitness cloud*.

Since high-fitness points tend to be much more important than low-fitness ones in determining the behaviour of evolutionary algorithms, an alternative algorithm to generate fitness clouds was proposed in [22]. The main steps of this algorithm can be informally summarised as follows: (1) Generate a set of individuals $\Gamma = \{\gamma_1, ..., \gamma_n\}$ by sampling the search space and let $f_i = f(\gamma_i)$, where $f(.)$ is the fitness function. (2) For each $\gamma_j \in \Gamma$ generate k neighbours, $v_1^j, \ldots, v_k^j$, by applying a genetic operator to γ_j and let $f'_j = \max_j f(v_j)$. (3) Finally, take $C = \{(f_1, f'_1), \ldots, (f_n, f'_n)\}$ as the fitness cloud. This is the interpretation of fitness cloud we will use in this paper. Note how this algorithm essentially corresponds to the sampling produced by a set of n stochastic hill-climbers at their first iteration after initialisation.

The fitness cloud can be of help in determining some characteristics of the fitness landscape related to evolvability and problem difficulty, but the mere observation of the scatterplot is not sufficient to quantify these features. The Negative Slope Coefficient *nsc* has been defined to capture with a single number some interesting characteristics of fitness clouds. It can be calculated as follows: let us partition C into a certain number of separate ordered "bins" $C_1, \ldots, C_m$ such that $(f_a, f'_a) \in C_j$ and $(f_b, f'_b) \in C_k$ with $j < k$ implies $f_a < f_b$. Consider the averages fitnesses $\bar{f}_i = \frac{1}{\lfloor |C_i| \rfloor} \sum_{(f,f') \in C_i} f$ and $\bar{f}'_i = \frac{1}{|C_i|} \sum_{(f,f') \in C_i} f'$. The points $(\bar{f}_i, \bar{f}'_i)$ can be seen as the vertices of a polyline, which effectively represents the "skeleton" of the fitness cloud. For each of the segments of this we can define a *slope*, $S_i = (f'_{i+1} - f'_i)/(f_{i+1} - f_i)$. Finally, the *nsc* is defined as:

$$nsc = \sum_{i=1}^{m-1} \min\,(0, S_i). \tag{1}$$

The hypothesis proposed in [22] is that *ncs* should classify problems in the following way: if $nsc = 0$, the problem is easy; if $nsc < 0$ the problem is difficult and the value of *nsc* quantifies this difficulty: the smaller its value, the more difficult the problem. The justification put forward for this hypothesis was that the presence of a segment with negative slope would indicate a bad evolvability for individuals having fitness values contained in that segment as neighbours would be, on average, worse than their parents in that segment [21]. (We will discuss an alternative, but related, justification for *nsc* later in this paper.)

The definition of *nsc* is very general and has many degrees of freedom. In particular, a question must be answered to be able to calculate the *nsc*: how should we partition the abscissas of a fitness cloud into bins? In [22], fitness clouds were statically partitioned into bins of the same size. Although this method is arbitrary, the results reported in [22] were encouraging and confirmed that the *nsc* is a suitable hardness indicator for many well-known GP benchmarks, including various versions of the symbolic regression problem, the even parity problem of many different orders and the artificial ant problem on the Santa Fe trail (all these benchmarks are documented in [13]). Furthermore, the *nsc* proved suitable also for some synthetic GP problems, such as GP Trap Functions [4], Royal Trees [17] and the Max problem [8], which naturally capture some typical features of easy and difficult fitness landscapes.

In [23] some limitations of the standard partitioning technique were pointed out and it was shown that it can generate bins that contain too few points. Furthermore, it was empirically shown that the *nsc* with this partitioning technique is *not* able to correctly predict the difficulty of two well known GP benchmarks: the multiplexer problem and the intertwined spirals problem [13]. To overcome these limitations, an alternative partitioning technique, *size driven bisection*, was proposed inspired by the well-known bisection algorithm. With this modification, *nsc* was shown to be a suitable hardness indicator for all the GP problems on which it had been tested, included the multiplexer and the intertwined spirals problems.

However, even *nsc* with size-driven bisection still lacks formality: e.g., a threshold for the minimum number of points belonging to a bin and the minimum admissible bin width have to be chosen arbitrarily. Even more importantly, still a formal justification of the *nsc* approach is missing.

In this paper, we propose a modified definition of *nsc*, the *fitness-proportional negative slope coefficient* (*fpnsc*), which we believe has the same qualities as the original *nsc* with the added bonus that it has a clear theoretical justification. The new coefficient has exactly the same definition of *nsc*, except for one thing: *we create fitness clouds using fitness proportional selection.* That is, for each individual $\gamma_j \in \Gamma$ we still generate k neighbours, $v_1^j, \ldots, v_k^j$, by applying a genetic operator. However, in *fpnsc* we then select a v_j using fitness proportionate selection and set $f'_j = f(v_j)$ instead of setting $f'_j = \max_j f(v_j)$ as is done in *nsc*.

3. MODELLING FPNSC

Let F be a stochastic variable representing the fitness of an individual randomly drawn from the population. If $\Phi(f)$ is the proportion of individuals in a population having fitness f, we have $E[F] = \sum_f f\Phi(f)$. Let us assume that the offspring are obtained by a unary operator, that, for simplicity, we will call mutation, and let F' be a stochastic variable representing the fitness of the offspring. Let $p_n(f'|f)$ be the

probability that the offspring of an individual of fitness f created by mutation have fitness f'. Then it is possible to compute the fitness distribution after a round of mutation

$$\Phi(f') = \sum_f p_n(f'|f)\Phi(f) \quad (2)$$

and also the expected value of the fitness of the offspring:

$$\begin{aligned} E[F'] &= \sum_{f'} f'\Phi(f') \\ &= \sum_{f'} f' \sum_f p_n(f'|f)\Phi(f) = \sum_f \Phi(f) \sum_{f'} f' p_n(f'|f). \end{aligned}$$

By definition of conditional expected value we have that $\sum_{f'} f' p_n(f'|f) = E[F'|f]$ and, so, $E[F'] = \sum_f E[F'|f]\Phi(f)$.

Note that the function $E[F'|f]$ tells us how the fitness of the offspring varies as a function of the fitness of the parents. So, its definition is very similar to that of the curve connecting the offspring-fitness-distribution centroids in the definition of *fpnsc*. In fact, $E[F'|f]$ is exactly the polyline one would obtain if selection was not used when computing the *fpnsc* (if the bins are sized in such a way that each one only includes points with the same parental fitness).

However, the *fpnsc* is based on the notion of first generating some neighbours of a point and then applying proportional selection on them. So, to provide a more precise interpretation of the *fpnsc* we need to add selection to our model. We do so by applying selection to the mutants of each individual. This is a local form of selection, known as soft brood selection. In the model we will assume that *all* the neighbours of all individuals of a particular fitness f take part in the selection step. This corresponds to taking the limit $k \to \infty$ which effectively is an infinite population assumption.[1]

In fitness proportionate selection, the selection probability for individuals of a particular fitness f is given by

$$p(f) = \left(\frac{f}{E[F]}\right)\Phi(f).$$

So, if by $p_s(f'|f)$ we denote the probability that the offspring (after mutation and selection) of an individual of fitness f will have fitness f', we have

$$p_s(f'|f) = \left(\frac{f'}{E[F'|f]}\right) p_n(f'|f)$$

where we used $E[F'|f]$ instead of $E[F']$ and $p_n(f'|f)$ instead of $\Phi(f')$ to account for the fact that selection is local (conditional) to the neighbours of a parent of fitness f.

So, if F'' is a stochastic variable representing the fitness of the offspring after mutation and selection, and $E[F''|f]$ is the conditional expected value of F'', we take as our *model* of *fpnsc* the following equation:

$$fpnsc = \sum_i \min\left(0, \frac{E[F''|f_{i+1}] - E[F''|f_i]}{f_{i+1} - f_i}\right). \quad (3)$$

[1]The infinite population assumption is a standard tool in the theory of evolutionary algorithms. This is an essential assumption if we wish to separate the intrinsic algorithm biases from the bias due to genetic drift.

Note that $E[F''|f]$ can be computed as follows

$$\begin{aligned} E[F''|f] &= \sum_{f''} f'' p_s(f''|f) \\ &= \sum_{f''} f'' \left(\frac{f''}{E[F'|f]}\right) p_n(f''|f) \\ &= \frac{E[F'^2|f]}{E[F'|f]}. \end{aligned}$$

From the well-known relation

$$Var[X] = E[(X - E[X])^2] = E[X^2] - (E[X])^2$$

we then obtain

$$E[F''|f] = \frac{Var[F'|f] + (E[F'|f])^2}{E[F'|f]} = \frac{Var[F'|f]}{E[F'|f]} + E[F'|f] \quad (4)$$

which indicates that, under (localised) fitness proportionate selection, the improvement in mean fitness of the offspring after selection is given by the ratio between variance and mean of the fitness of the offspring *before* selection. It is clear that whether or not the mean fitness after mutation and selection, $E[F''] = \sum_f E[F''|f]\Phi(f)$, is going to be higher or lower than $E[F] = \sum_f f\Phi(f)$ depends on the shape of the function $E[F''|f]$ and whether or not this is above or below the diagonal $f'' = f$, particularly in densely populated areas of the fitness distribution $\Phi(f)$ (e.g., near the mean parental fitness). More discussion on the interpretation of *fpnsc* is provided in Section 6.

As a simple example, let us consider an (unrealistic) problem and a mutation operation where each individual of fitness f has two equally-likely neighbours: one with fitness $f+1$, the other with fitness $f-1$. Therefore,

$$p_n(f'|f) = \begin{cases} 1/2 & \text{if } f' = f+1, \\ 1/2 & \text{if } f' = f-1, \\ 0 & \text{otherwise,} \end{cases}$$

and

$$E[F'|f] = \sum_{f'} f' p_n(f'|f) = (f+1)/2 + (f-1)/2 = f.$$

With the value of $E[F'|f]$ in hand, we can then compute

$$\begin{aligned} Var[f'|f] &= \sum_{f'} (f' - f)^2 p_n(f'|f) \\ &= \frac{(f+1-f)^2}{2} + \frac{(f-1-f)^2}{2} = 1. \end{aligned}$$

So, the expected fitness of the offspring after mutation and selection is given by $E[F'''|f] = 1/f + f$. This shows that as the parental fitness increases, the mean fitness of the individuals produced with a mutation/selection step progressively approaches the average offspring fitness (without selection). This makes sense, since the selection pressure exerted by proportional selection progressively decreases in these circumstances. Also, assuming $f \geq 1$ and that the fitness-cloud bins are of the form $C_i = \{(i, i-1), (i, i+1)\}$, the slope of the $E[F''|f]$ curve

$$S_i = \frac{1/(i+1) + (i+1) - 1/i - i}{i+1-i} = \frac{i(i+1) - 1}{i(i+1)}$$

is always positive, and, so, $fpnsc = \sum_i \min(0, S_i) = 0$, suggesting that this problem would be easy.

3.1 Related work

In [24], Verel studied fitness clouds and computed an explicit expression [24, page 26] for the function $E[F'|f]$ for a particular set of functions, namely the *uniform embedded* (UE) functions, and for a mutation operator that generates offspring by sampling uniformly at random among the strings of a given Hamming distance d from the parent solution. UE functions – a subset of the embedded functions defined in [10] – generalise the NK and MAX-SAT landscapes. Verel's results generalised previous results for NKp and NKq landscapes presented in [19] and for NKp landscapes presented in [2].

Verel also considered [24, page 35] the conditional expected fitness for the offspring generated by a hill-climber, a quantity we called $E[F''|f]$, for the same landscapes and mutation operator mentioned above. No explicit expression for $E[F''|f]$ was obtained, and the study of the hill-climber for UE functions was empirical. The particular hill-climber considered was deterministic since it was based on an extreme form of selection where all valid neighbours of an individual are generated and the one with the highest fitness is selected as offspring.

These results are important and relevant to our work, but they are orthogonal to ours. For example, they do not apply to the invertible functions of unitation, to standard GA mutation and to the more typical forms of selection that are considered in this paper. Also, since an explicit formulation for $E[F''|f]$ was not obtained, Verel's results were not used to give a formal interpretation and assessment of the *nsc*, while this is the main objective of this paper.

Our work has also some similarities with the work presented in [5, 26], where the (1+1) EA's behavior is modelled for a particular subset of separable functions. Also, in [18], the fixed points of a simple GA were theoretically obtained for some functions of unitations similar to the ones that will be studied later in this paper. Finally, our work has a strong relation with [27], where the behavior of simple GAs for some functions of unitation was studied, even though the goal of [27] was not the formalization or theoretical assessment of the *nsc*.

4. INVERTIBLE FUNCTIONS OF UNITATION UNDER ONE-BIT MUTATION

Let us use one-bit mutation as our search operation. Under this mutation operator the offspring is produced by flipping the bit at a randomly chosen locus in the parental string. So, mutants are Hamming distance 1 from parents.

Let us consider a generic function of unitation, $f(u)$, that is *invertible*. Let h be the inverse of f. This means that it is possible to compute the unitation value, u, of an individual from its fitness, $\tilde{f}$, i.e., $u = h(\tilde{f})$. It is then possible to compute the fitness of the mutants of an individual of fitness $\tilde{f}$, namely $f^+ = f(h(\tilde{f})+1)$ and $f^- = f(h(\tilde{f})-1)$. How many neighbours of an individual have fitness f^+ vs. fitness f^- depends on the unitation value of the parent. Namely, there are $h(\tilde{f})$ neighbours with fitness f^- and $\ell - h(\tilde{f})$ neighbours with fitness f^+. More precisely, it is easy to see that

$$p_n(f'|f) = \frac{h(f)}{\ell}\delta(f' = f(h(f)-1)) + \frac{\ell - h(f)}{\ell}\delta(f' = f(h(f)+1)),$$

Table 1: Performances of a GA using one-bit mutations on different functions of unitation and two values of bit-string length.

	Onemax	Trap	Onemix
$\ell = 10$	1	0.1	0.1
$\ell = 100$	1	0	0

where $\delta(x)$ is 1 if x is true, 0 otherwise. Therefore,

$$\begin{aligned} E[F'|f] &= \sum_{f'} f' p_n(f'|f) \\ &= f(h(f)-1)\frac{h(f)}{\ell} + f(h(f)+1)\frac{\ell - h(f)}{\ell} \end{aligned}$$

By applying effectively the same steps in the previous calculation to the stochastic variable $(F')^2$ instead of F', we obtain

$$\begin{aligned} & E\left[(F')^2 |f\right] \\ &= (f(h(f)-1))^2 \frac{h(f)}{\ell} + (f(h(f)+1))^2 \frac{\ell - h(f)}{\ell}. \end{aligned}$$

Taking the ratio between the last two results, we can compute the expected fitness of the offspring after selection:

$$E[F''|f] = \frac{(f(h(f)-1))^2 h(f) + (f(h(f)+1))^2 (\ell - h(f))}{f(h(f)-1)h(f) + f(h(f)+1)(\ell - h(f))}.$$

If we apply the calculation to Onemax for $\ell = 10$ (see Fig. 1(a)), we obtain the plot shown in Fig. 1(d), which has always positive slope. As a result $fpnsc = 0$, indicating an easy problem. If, instead, we apply the calculations to the deceptive trap function in Fig. 1(b), we obtain the plot in Fig. 1(e), which has two sections with negative slope. In this case $fpnsc = -10.568$, indicating, according to the *fpnsc* hypothesis, that the problem is harder. Finally, if we calculate $E[F''|f]$ with selection for the Onemix function [16] shown in Fig. 1(c) we obtain the plot in Fig. 1(f). The latter has negative slope in 6 segments out of 10, and $fpnsc = -12.257$, indicating that the problem is slightly harder than Trap under one-bit flip mutation.

In order to experimentally confirm these results, we used a simple GA with the following characteristics: one-bit mutation was applied to each individual at each generation, no crossover was used, populations were of size 100, tournament selection with tournament size 10 was used, and the maximum number of generations was 100. We defined as our performance measure the fraction of runs in which the global optimum was found by generation 100. To assess this, we performed 100 independent runs. Tab. 1 shows the success rates obtained in our experiments with different fitness functions and string lengths. With this form of mutation the GA is able to find solutions to the Onemix and Trap problems effectively only if they happen to be in the population at generation 0. On the contrary, Onemax is consistently solved. This confirms the *fpnsc* predictions.

5. INVERTIBLE FUNCTIONS OF UNITATION UNDER STANDARD MUTATION

Let us consider again a generic invertible function of unitation, $f(u)$, with inverse $h(f)$. However, this time, let us

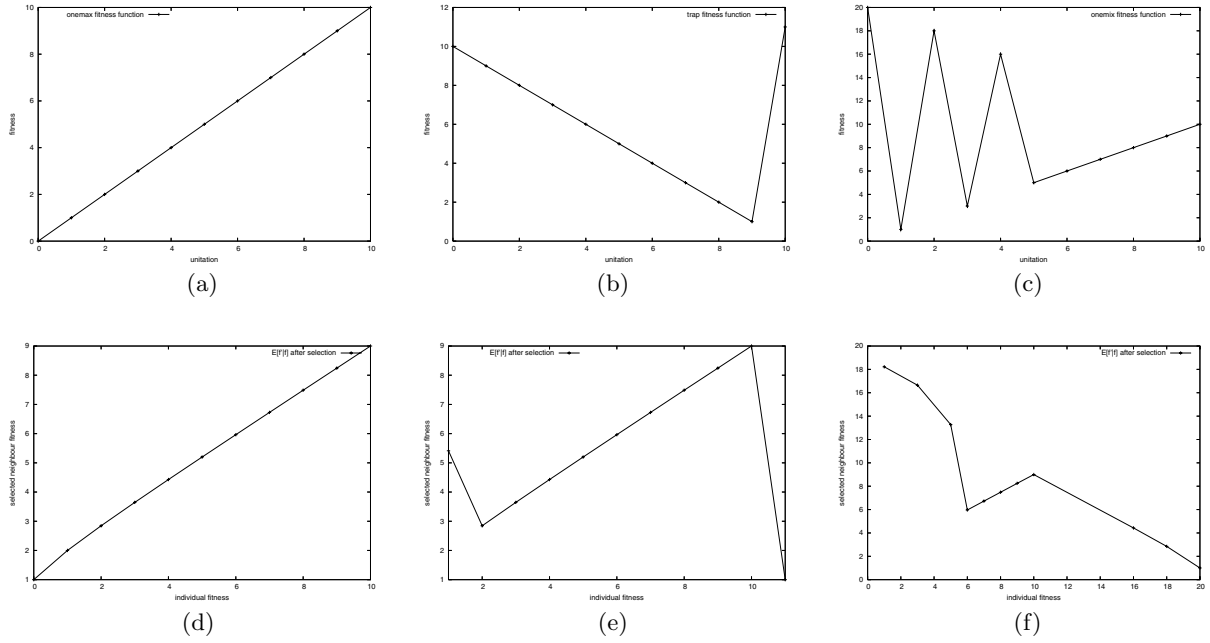

Figure 1: (a) Onemax, (b) Trap, (c) Onemix, (d) plot of $E[F''|f]$ for Onemax, (e) plot of $E[F''|f]$ for Trap, and (f) plot of $E[F''|f]$ for Onemix.

assume that we use standard bit-flip mutation with a mutation rate (per bit) p_m.

An individual with unitation class u will have u bits set to 1 and $\ell - u$ bits set to 0. Let us imagine that we create the offspring by going through the bits that are 0 and flipping a biased coin to decide whether to mutate them or not. Let Z_0 be a stochastic variable that describes the number of 0's mutated into 1's and let Z_1 be the number of 1's mutated into 0's. Naturally, both Z_0 and Z_1 are binomially distributed, with success probability p_m. So

$$\Pr(Z_0 = z) = \binom{\ell - u}{z} p_m^z (1 - p_m)^{\ell - u - z}$$

and

$$\Pr(Z_1 = z) = \binom{u}{z} p_m^z (1 - p_m)^{u - z}.$$

The unitation value of the offspring produced by bit-flip mutation will then be described by the stochastic variable $U = u + Z_0 - Z_1$, the distribution of which can be computed as follows

$$\begin{aligned} \Pr(U = y) &= \Pr(Z_0 - Z_1 = y - u) \\ &= \sum_{z_1=0}^{u} \Pr(Z_1 = z_1) \Pr(Z_0 = z_1 + y - u) \end{aligned}$$

It is then easy to show that

$$\begin{aligned} \Pr(U = y) &= \sum_{z_1=\max(0,u-y)}^{\min(u,\ell+u-y)} \binom{u}{z_1} \binom{\ell - u}{z_1 + y - u} \\ &\times \; p_m^{(2z_1+y-u)} (1 - p_m)^{\ell-(2z_1+y-u)} \end{aligned}$$

Let $\mathcal{F}(f) = \{f_0, f_1, \cdots, f_\ell\}$ be the set of codomain values associated to the unitation values $u = 0, 1, \cdots, \ell$. That is, $f_0 = f(0)$, $f_1 = f(1)$, etc. Note that these are all distinct, since f is invertible. Then we can write

$$p_n(f'|f) = \begin{cases} \Pr(U = h(f')) & \text{if } f' \in \mathcal{F}(f), \\ 0 & \text{otherwise,} \end{cases}$$

or, more explictly,

$$\begin{aligned} p_n(f'|f) &= \delta(f' \in \mathcal{F}(f)) \sum_{z=\max(0,h(f)-h(f'))}^{\min(h(f),\ell+h(f)-h(f'))} \binom{h(f)}{z} \\ &\times \binom{\ell - h(f)}{z + h(f') - h(f)} p_m^{(2z+h(f')-h(f))} \\ &\times (1 - p_m)^{\ell-(2z+h(f')-h(f))} \end{aligned}$$

By applying the definition of conditional expected value, one can then easily prove that

$$\begin{aligned} E[F'|f_j] &= \sum_{i=0}^{\ell} f_i \sum_{z=\max(0,j-i)}^{\min(j,\ell+j-i)} \binom{j}{z} \binom{\ell - j}{z + i - j} p_m^{(2z+i-j)} \\ &\times (1 - p_m)^{\ell-(2z+i-j)} \end{aligned}$$

and

$$\begin{aligned} E\left[(F')^2 \,\middle|\, f_j\right] &= \sum_{i=0}^{\ell} f_i^2 \sum_{z=\max(0,j-i)}^{\min(j,\ell+j-i)} \binom{j}{z} \binom{\ell - j}{z + i - j} \\ &\times p_m^{(2z+i-j)} (1 - p_m)^{\ell-(2z+i-j)} . \end{aligned}$$

The ratio between the last two results gives us the expected fitness of the offspring after selection, $E[F''|f]$.

If we apply the calculation to the Onemax function for $\ell = 10$ (Fig. 1(a)), we obtain the plots shown in Fig. 2. These have always positive slope. As a result, as shown in

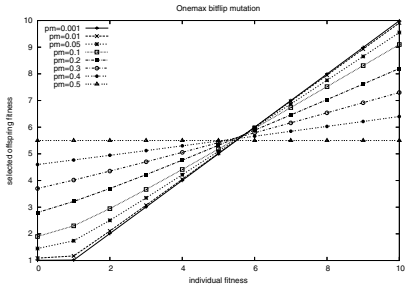

Figure 2: Plot of $E[F''|f]$ for Onemax for different mutation probabilities.

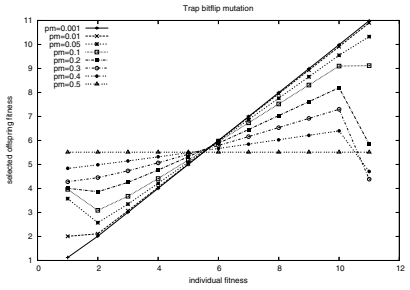

Figure 3: Plot of $E[F''|f]$ for Trap for different mutation probabilities.

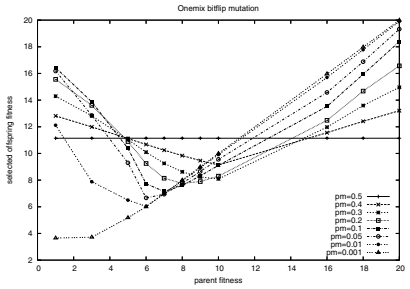

Figure 4: Plot of $E[F''|f]$ for Onemix for different mutation probabilities.

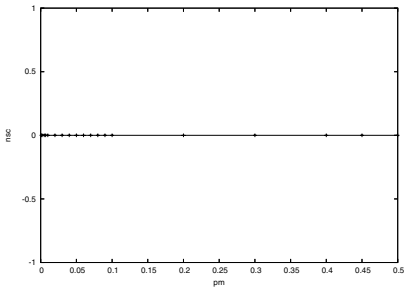

Figure 5: Plot of *fpnsc* for Onemax for different values of mutation probability.

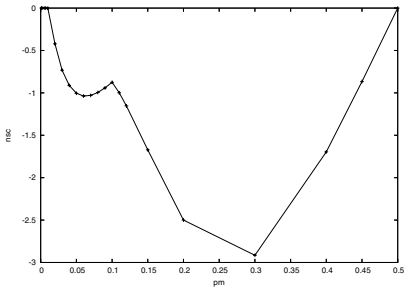

Figure 6: Plot of *fpnsc* for Trap for different values of mutation probability.

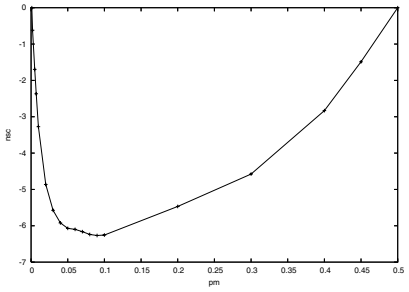

Figure 7: Plot of *fpnsc* for Onemix for different values of mutation probability.

Fig. 5, *fpnsc* is zero irrespective of the value of p_m, indicating an easy problem.

If, instead, we apply the calculations to the deceptive trap function (Fig. 1(b)), we obtain the plots in Fig. 3. Unlike for Onemax, here the the plot of $E[F''|f]$ changes significantly with the mutation rate, showing two sections with negative slopes for most mutation rates except very high and very low ones. So, as shown in Fig. 6, *fpnsc* is non-zero, indicating a harder problem. Also *fpnsc* varies slightly with the mutation rate, suggesting that the problem difficulty may be maximum at mutation rates of around 0.3.

Finally, if we calculate $E[F''|f]$ for the Onemix function (Fig. 1(c)) we obtain the plots in Fig. 4 and Fig. 7. Clearly, this problem presents several segments with negative slope (except for very low and very high mutation rates). As a result the values of *fpnsc* are significantly more negative than the corresponding values for Trap, for most values of p_m. This would suggest that the problem is harder than Trap. Also, note that the *fpnsc* values are higher than the value recorded with one-bit mutations (namely, -10.568). This suggests that with bit-flip mutation the problem is not as hard. The difference is due to the right hand side of the offspring fitness plots, where, with bit flip mutation, the slope is positive. This is due to the fact that, with this mutation, it is always possible (albeit with a probability depending on p_m) for the offspring of an individual at one of the fitness peaks on the left of Fig. 1(c) to jump on another fitness peak without having to go through a low-fitness ditch (as is the case for one-bit mutation). Finally, note that in Fig. 7, *fpnsc* vary widely with the mutation probability, reaching its minimum for values of p_m between 0.03 and 0.1. Again, this would suggest that the problem is hardest at those mutation rates.

We should note that for all problems, the offspring fitness vs. parent fitness plots tend to coincide with the main diagonal of the diagram whenever $p_m \approx 0$. This is because, when the mutation probability is very small, the offspring tend to be exact copies of the parents. So, their fitness is on average very close to the parental fitness. It is also worth noting that when $p_m \approx 0.5$ the offspring are effectively random strings,

Table 2: Performances of a GA with a population of 100 individuals, run for 100 generations with a tournament size 10, for various values of ℓ and p_m.

	Onemax	Trap	Onemix
$\ell = 10$, $p_m = 0.1$	1	0.4	1
$\ell = 10$, $p_m = 0.01$	1	0.11	0.65
$\ell = 10$, $p_m = 0.001$	1	0.1	0.1
$\ell = 100$, $p_m = 0.1$	1	0	0
$\ell = 100$, $p_m = 0.01$	1	0	0
$\ell = 100$, $p_m = 0.001$	1	0	0

irrespective of what parent they came from. So, the average fitness of the offspring becomes independent of the parental fitness, leading to a flat offspring fitness vs. parent fitness curve. As a result, in both these cases the curves present no negative slopes and $fpnsc = 0$. So, one should not expect the *fpnsc* figure to be a good predictor of problem difficulty whenever p_m is either very small or very large.

Tab. 2 reports the success rate for runs with populations of 100 individuals evolved for 100 generations with a tournament size of 10. Results are means over 100 independent runs. Naturally, the process of sampling taking place in a genetic algorithm or GP system is very different from the sampling process used in the definition of *fpnsc*. So, we should not expect to obtain a one-to-one correspondence between *fpnsc* and performance. However, *fpnsc* correctly predicted that Onemix and Trap would be hard while Onemax would be easy. Also, *fpnsc* predicted that Onemix would become easier as the mutation rate increases while Onemax would remain easy irrespective of the mutation rate. In addition, *fpnsc* predicted that both Trap and Onemix would be harder to solve using one-bit mutations than standard bit-flip mutations. However, *fpnsc* incorrectly predicted that Trap is easier than Onemix with bit-flip mutation.

To emphasise these effects for the case of $\ell = 10$, where the search space includes only 1024 strings, we performed a larger set of runs (1000 independent runs for each mutation rate and problem) with a smaller population (10 individuals) and a reduced selection pressure (tournament size 2). Also the maximum number of generations was reduced to 30. The success rates recorded in these runs are shown in Fig. 8. These results confirm the previous observations. In particular, we should note that, except for unreasonably low mutation rates, Onemax is solved with a much higher success rate than what is expected from random search (around 30% for random search without resampling, a bit less for random search with resampling) if one performed the same number of trials as our GA. On the contrary, irrespective of the mutation rate, Onemix and Trap are consistently hard for our GA, being its performance well below that of random search.

6. DISCUSSION

In Section 3 we explicitly formalised the relation between the curve, $E[F''|f]$, used to compute the *fpnsc*, and the conditional mean and variance in the fitness of the offspring produced by mutation. In Sections 4 and 5 we then applied this relation to compute $E[F''|f]$ for two mutation operators and for *all* invertible functions of unitation. This allowed us, for the first time, to compute *fpnsc* exactly and from

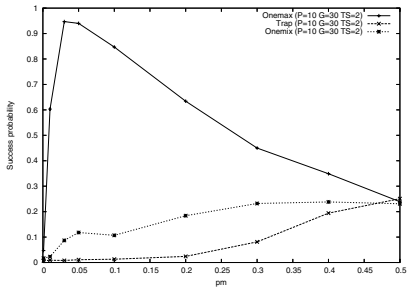

Figure 8: Plot of the success probability for runs of a simple GA on Onemax, Trap and Onemix for different values of mutation probability, a population of 10 individuals, evolved for 30 generations with a tournament size 2.

first principles rather than using sampling. In the same two sections we also saw how the *fpnsc* appears to be quite a reliable classifier for problem hardness, although, being an unnormalised integral measure, the *fpnsc* does not appear to have the resolution necessary to predict fine differences in problem hardness.

In this section we want to discuss what may be the reasons for the good reliability of the *fpnsc* and see if there are ways of improving it further.

Whenever the curve $E[F''|f]$ falls under the diagonal ($f'' = f$), the average fitness of the offspring of an individual is lower than the fitness of the parent. So, the difference $E[F''|f] - f$ indicates the absolute evolvability of a genotype. Clearly it is unavoidable that at least in some part of the $E[F''|f]$ curve evolvability will be low, e.g., near the global optimum. However, it would appear that if high fitness values are the objective, any situation where the offspring are less fit than their parents will be an obstacle to the progress of the population towards high fitness values. So, in principle depressions in the $E[F''|f]$ curve may be indicators of potential problem difficulty *for a hill-climber*.

However, in a GA, whether or not a genotype and its offspring can reproduce successfully depends not only on their absolute evolvability but also on what else is competing for survival in the population. Typically, because of the selection phase, phenotypic (fitness) diversity will be limited and, therefore, competitors will be all densely distributed around a particular fitness value. If the $E[F''|f]$ curve presents a negative slope at that particular value, we have the situation that the offspring of above average fitness parents will be less fit than the offspring of below average fitness parents. So, over two generations we expect a slower increase, a stagnation, or even a lowering of the mean fitness of the population. If the slope was positive instead, we would expect rapid progress towards high mean fitness values. So, negative-slope segments in $E[F''|f]$ are obstacles to the progress of the population towards high fitness values, and this is why the *fpnsc* is an indicator of problem difficulty for GAs.

7. CONCLUSIONS

The negative slope coefficient is an empirical measure of problem hardness based the analysis of offspring-fitness vs. parent-fitness scatterplots. The *nsc* has been tested empirically on a large variety of genetic programming problems

showing considerable reliability in evaluating problem hardness. However, no theoretical investigation or explanation for *nsc* has ever been proposed. In this paper, by slightly modifying the way in which *nsc* is computed, we have obtained a new hardness measure, the fitness-proportional *nsc*, which is amenable to theoretical analysis. In particular, we have theoretically clarified what exactly *fpnsc* computes and evaluated the applicability of the *fpnsc* to GAs operating on fixed-length strings. More specifically, we computed the *fpnsc* theoretically for the class of invertible functions of unitation, and for two mutation operators. We then applied the theory to compute the *fpnsc* for three benchmark functions – Onemax, Trap and Onemix – and compared the predictions of the *fpnsc* with the success probability recorded in actual runs.

The results suggest that the *fpnsc* is able to broadly discriminate between easy and hard GA problems, although it appears also to be unable to pick up small differences in problem hardness. In future work we hope to be able to overcome this limitation.

REFERENCES

[1] L. Altenberg. The evolution of evolvability in genetic programming. In K. Kinnear, editor, *Advances in Genetic Programming*, pages 47–74, Cambridge, MA, 1994. The MIT Press.

[2] L. Barnett. *Evolutionary Search on Fitness Landscapes with Neutral Networks*. PhD thesis, University of Sussex, 2003.

[3] P. Collard, S. Verel, and M. Clergue. Local search heuristics: Fitness cloud versus fitness landscape. In R. L. D. Mántaras and L. Saitta, editors, *2004 European Conference on Artificial Intelligence (ECAI04)*, pages 973–974, Valence, Spain, 2004. IOS Press.

[4] K. Deb and D. E. Goldberg. Analyzing deception in trap functions. In D. Whitley, editor, *Foundations of Genetic Algorithms, 2*, pages 93–108. Morgan Kaufmann, 1993.

[5] S. Droste, T. Jansen, and I. Wegener. A rigorous complexity analysis of the (1 + 1) evolutionary algorithm for separable functions with boolean inputs. *Evolutionary Computation*, 6(2):185–196, 1998.

[6] C. Flamm, I. L. Hofacker, P. F. Stadler, and M. T. Wolfinger. Barrier trees of degenerate landscapes. *Z. Phys. Chem.*, 216:155–173, 2002.

[7] S. Forrest and M. Mitchell. What makes a problem hard for a genetic algorithm? some anomalous results and their explanation. *Machine Learning*, 13:285–319, 1993.

[8] C. Gathercole and P. Ross. An adverse interaction between crossover and restricted tree depth in genetic programming. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 291–296, Stanford University, CA, USA, 28–31 July 1996. MIT Press.

[9] J. Hallam and A. Prügel-Bennett. Large barrier trees for studying search. *IEEE Trans. Evolutionary Computation*, 9(4):385–397, 2005.

[10] R. B. Heckendorn. Embedded landscapes. *Evolutionary Computation*, 10(4):345–369, 2002.

[11] J. Horn and D. E. Goldberg. Genetic algorithm difficulty and the modality of the fitness landscapes. In D. Whitley and M. Vose, editors, *Foundations of Genetic Algorithms, 3*, pages 243–269. Morgan Kaufmann, 1995.

[12] T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In L. J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 184–192. Morgan Kaufmann, 1995.

[13] J. R. Koza. *Genetic Programming*. The MIT Press, Cambridge, Massachusetts, 1992.

[14] M. Mitchell, S. Forrest, and J. Holland. The royal road for genetic algorithms: fitness landscapes and ga performance. In F. J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems, Proceedings of the First European Conference on Artificial Life*, pages 245–254. The MIT Press, 1992.

[15] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, Massachusetts, 1994.

[16] R. Poli, A. H. Wright, N. F. McPhee, and W. B. Langdon. Emergent behaviour, population-based search and low-pass filtering. In *2006 IEEE World Congress on Computational Intelligence, 2006 IEEE Congress on Evolutionary Computation*, pages 395–402, Vancouver, 16-21 July 2006.

[17] B. Punch, D. Zongker, and E. Goodman. The royal tree problem, a benchmark for single and multiple population genetic programming. In P. Angeline and K. Kinnear, editors, *Advances in Genetic Programming 2*, pages 299–316, Cambridge, MA, 1996. The MIT Press.

[18] J. E. Rowe. Population fixed-points for functions of unitation. In *Foundations of Genetic Algorithms, FOGA 1998*, pages 69–84, 1998.

[19] T. Smith, P. Husbands, P. Layzell, and M. O'Shea. Fitness landscapes and evolvability. *Evolutionary Computation*, 1(10):1–34, 2001.

[20] M. Tomassini, L. Vanneschi, P. Collard, and M. Clergue. A study of fitness distance correlation as a difficulty measure in genetic programming. *Evolutionary Computation*, 13(2):213–239, 2005.

[21] L. Vanneschi. *Theory and Practice for Efficient Genetic Programming*. Ph.D. thesis, Faculty of Science, University of Lausanne, Switzerland, 2004. Downlodable version at: http://www.disco.unimib.it/vanneschi.

[22] L. Vanneschi, M. Clergue, P. Collard, M. Tomassini, and S. Vérel. Fitness clouds and problem hardness in genetic programming. In K. D. *et al.*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'04*, volume 3103 of *Lecture Notes in Computer Science*, pages 690–701. Springer, Berlin, Heidelberg, New York, 2004.

[23] L. Vanneschi, M. Tomassini, P. Collard, and S. Vérel. Negative slope coefficient. A measure to characterize genetic programming. In P. Collet, M. Tomassini, M. Ebner, S. Gustafson, and A. Ekárt, editors, *Proceedings of the 9th European Conference on Genetic Programming*, volume 3905 of *Lecture Notes in Computer Science*, pages 178–189, Budapest, Hungary, 10 - 12 Apr. 2006. Springer.

[24] S. Verel. *Étude et Exploitation des Réseaux de Neutralité dans les Paysages Adaptatifs pour l'Optimisation Difficile*. PhD thesis, University of Nice – Sophia Antipolis, France, 2005. In French Language.

[25] S. Verel, P. Collard, and M. Clergue. Where are bottleneck in NK fitness landscapes? In *CEC 2003: IEEE International Congress on Evolutionary Computation. Canberra, Australia*, pages 273–280. IEEE Press, Piscataway, NJ, 2003.

[26] I. Wegener. On the expected runtime and the success probability of evolutionary algorithms. In *Workshop on Graph-Theoretic Concepts in Computer Science*, pages 1–10, 2000.

[27] A. H. Wright and J. N. Richter. Strong recombination, weak selection, and mutation. In *Proceedings of the 8th annual conference on Genetic and Evolutionary Computation, GECCO 2006*, pages 1369–1376, 2006.

[28] S. Wright. The roles of mutation, inbreeding, crossbreeding and selection in evolution. In D. F. Jones, editor, *Proceedings og the Sixth International Congress on Genetics*, volume 1, pages 356–366, 1932.