Dealings with Problem Hardness in Genetic Algorithms

STJEPAN PICEK
Ring Datacom d.o.o.
Trg J. J. Strossmayera 5, 10000 Zagreb
CROATIA
stjepan@ring.hr

MARIN GOLUB
Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb
CROATIA
marin.golub@fer.hr

Abstract: Genetic algorithms (GA) have been successfully applied to various problems, both artificial as well as real-world problems. When working with GAs it is important to know those those kinds of situations when they will not find the optimal solution. In other words, to recognize problems that are difficult for a GA to solve. There are various reasons why GAs will not converge to optimal solutions. By combining one or more of these reasons a problem can become a GA-hard problem. Today, there are numerous methods for solving GA-hard problems; every measure has its specific advantages and drawbacks. In this work the effectiveness of one of these measures is evaluated, namely the Negative Slope Coefficient (NSC) measure. A different measure is proposed, called the New Negative Slope Coefficient (NNSC) measure, which aims to address certain drawbacks of the original method. Possible guidelines for further development of this, and comparable methods are proposed.

Key–Words: Genetic Algorithm, Unitation, Fitness Landscape, Negative Slope Coefficient, Hardness, Difficulty, Deception

1 Introduction

Genetic algorithms (GA) are successfully applied to a variety of problems, which include scheduling problems [4], control optimization [23] and forecasting applications [11]. Besides these successes, they have also demonstrated poor results. Considering the widespread use of genetic algorithms it is necessary to know in which cases they will be successful, and in which they will not find an optimal solution. In other words, to find out which problems are difficult for them to solve. Based on the works of Bethke (1980) and Goldberg (1987), the term 'deception' has been introduced in order to understand the situations when a GA might fail. These deceptive functions create a problem for the genetic algorithm when performing optimization tasks. Currently a number of reasons are known that address the problems that are difficult for a GA to solve. One important concept is the Fitness Landscape. The Fitness Landscape presents a powerful metaphor for global optimization. It presents a visualization of the link between the genotype or phenotype in the given population and their respective probability of reproduction.

It is impossible to define a practical fitness landscape because the solution space is simply too large. Instead researchers have been looking for ways of defining interesting characteristics of a fitness landscape. Among others, a measure of the ruggedness of a fitness landscape is worth mentioning, as is reflected in the work of Weinberger, Jones and Forrest, and Vanneschi. Weinberger introduced the autocorrelation function and the correlation length for random walks. Jones and Forrest proposed the fitness distance correlation (FDC) - the correlation of the fitness of an individual and its distance from the global optimum. FDC presents a very reliable measure of problematic difficulty for a GA. However, it has several shortcomings. Among others, it is important to know the optimal solution upfront. Vanneschi et al. first introduced the Negative Slope Coefficient measure (NSC measure), which could be considered as an extension of Altenberg's evolvability measure [28]. Vanneschi also introduced the Fitness-Proportionate Negative Slope Coefficient measure as a supplement to his NSC method. An advantage of these methods is that they are predictive, that is, it is not necessary to know an optimal solution prior to an experiment itself [27].

In section 2, the basic concepts about deception and problem hardness in GAs are defined, in section 3 the necessary background information is presented that is needed to understand this work, section 4 defines the parameters used in the experiments, the performance measure for the comparison, experiments conducted on unitation functions and the results achieved for the NSC measure; section 5 repeats the experiments, but this time for the New NSC measure, and finally, section 6 draws a conclusion and future recommendations.

2 Deceptivity and Problem Hardness

2.1 Deception and Deceptivity

To be able to understand concept of deception, some definitions about genetic algorithms in general are needed. Whitley stated

> A primary hyperplane competition of order-N involves the complete set of primary competitors, where the primary competitors in a hyperplane competition of order-N are set of 2^N hyperplanes having schemata with N bit values in the same locations [29].

Besides this,

> Deception implies that the global winner of some hyperplane competition of order-N has a bit pattern that is different from the bit pattern of the "'global winner"' for some "'relevant"' lower order competition [29].

Deceptive functions are a family of fitness landscapes developed to challenge the building block hypothesis. The building block hypothesis is a direct result of schema theorem and it states that GAs work by combining low-order building blocks to form higher-order ones. Deceptive functions are an example of functions where lower-order building blocks do not form higher-order building blocks [16][30]. The term of deceptivity is quite general because most of the problems have some degree of deception. Therefore, it is important to try to distinguish possible deceptive situations. A deceptive problem is any problem of order-N that involves deception in one or more relevant lower order hyperplane competitions. The problem is fully deceptive when all the relevant lower order hyperplane competitions lead toward a deceptive attractor. It can be shown that for a fully deceptive problem, the deceptive attractor is the complement of a "'global winner"' of the hyperplane competition at order-N. A consistently deceptive problem of order-N occurs when none of relevant hyperplane competitions lead to "'global winner"' but all of the relevant hyperplane competitions do not necessarily lead toward the deceptive attractor (except in order-1 hyperplane competition). Deceptive functions therefore represent consistently deceptive problems where the number of bits used to encode the solution space match the order of deception. A problem is partially deceptive in some order-k if and only if all hyperplanes competition of order-(k-1) are deceptive [29]. It is not always easy to see the connection between deceptiveness and problem hardness because there are nondeceptive problems that are hard for a GA to solve. On the other side, there are deceptive problems that are easy for GA to solve. However, some deceptive problems are known to be consistently difficult for GA to solve. When deception is coupled with some other problem (like epistasis or bad linkage) then the overall problem will be hard for a GA to solve.

2.2 GA-Hard Problems and Complexity

When trying to define GA-hard problem, an obvious definition could be that those are the problems that have the ability to seriously mislead the GA search. Yet, a more strict definition of a GA-hard problem has never been formally defined. Reasons are, amongst others, that most of the researchers tend to find problem characteristics that cause a GA to fail to converge, but few attempt to consistently distinguish between a problem and a problem instance [22]. Using the techniques from complexity theory it is possible to define a GA-hard problem more formally [21][22]. First, a definition of a PO and NPO complexity classes will be given.

Definition 1 *The class PO is the class of optimization problems that can be solved in polynomial time with a Deterministic Turing Machine (DTM) [21].*

Definition 2 *The class NPO is the class of optimization problems that can be solved in polynomial time with a nondeterministic Turing Machine (NTM) [21].*

For a given complexity class, the formal way to define the hard problem is to show that every problem in the class reduces to the hard problem. In other words, a reduction R is a mapping from problem A onto problem B in such a way that one can optimize any problem instance x of A by optimizing B(R(x)). Before giving the definition of a GA-hard problem it is necessary to describe a method for evaluating a problem for GA. The goal for a representation is to minimize the number of bits in a chromosome that still uniquely identifies a solution to the problem.

Definition 3 *For a problem P, and Dn the set of instances of P of size n, let MCL(P,n) be the least l for which there is an encoding e:$S^1 \rightarrow$ Dn with a domain dependant evaluation function g, where g and e are in FP (the class of functions computable in polynomial time) [21].*

A more formal definition of a GA-hard problem is:

Definition 4 *A problem H is hard for class C if, for any problem A $\in$ C, and any instance x of A, optimizing R(A(x)) optimizes x [21].*

For those problems that do not fit the classical definition of hardness and yet are inherently less efficient than if solved using a method other than a GA a definition of a GA-semi-hard problem could be given.

Definition 5 *An optimization problem P is called Semi-hard if there is a DTM M that solves it in polynomial time and MCL(P,n) $\in$ O(n) [21].*

3 Background

This section addresses the information necessary for a complete understanding of the article. Here the terms, Fitness Landscape, Fitness Clouds, Unitation Functions that are used in the work, Metropolis-Hastings Sampling and the Negative Slope Coefficient measure, are explained.

3.1 Fitness Landscape

The fitness landscape can be defined as a search space (S), a metric and scalar fitness function defined on the elements of S. If it assumed that the goal is to maximize fitness, then the best global solutions are the 'peaks' in the search space. The local optima can also be defined as follows: assume a non-negative-real valued, scalar fitness function f(s) over a binary string s of the length l, where

$$f(s)\epsilon\Re \geq 0 \ . \tag{1}$$

Suppose that, generally speaking, 'f' has to be maximized. The local optimum in the discrete S search space is a point or an area, whose fitness function value is larger than all of its closest neighbours. A region is considered to be a neutral network if the linked points all have the same fitness. In other words, a set of points in proximity of the nearest-neighbour points of equal fitness are considered as a single optimum. For a move operator in search space, bit mutation is used. The 'nearest neighbour' is a measure that shows the distance between points s_1 and s_2, where s_1 and s_2 represent two binary strings. The Hamming distance has used to determine the distance (i.e. the number of bit positions in which two binary strings differ) [8].

3.2 Fitness Clouds

Let $\gamma = (\gamma_1, \gamma_2, ..., \gamma_n)$ represent the entire search space of a GA problem and $V(\gamma)$ be a set of all the neighbours of the $\gamma\epsilon\Gamma$ individual, which is obtained by the application of a standard bit-flip mutation. The choice of neighbors is a result of a tournament selection method with k=10 being the selection parameter. An individual with the highest fitness value is picked up as a neighbor. The following set of points at the bi-dimensional plane is defined:

$$P = \{(f(\gamma), f(\nu)), \forall\gamma\epsilon\Gamma, \forall\nu\epsilon V(\gamma)\} \ . \tag{2}$$

The P diagram is a scatterplot fitness of the values of all the individuals that belong to the search space versus their neighbors' fitness. A fitness cloud implicitly gives an insight into a genotype against the phenotype mapping [26].

Metropolis-Hastings Sampling Generally, the search space is too big in order to consider all the individuals. Therefore samples are taken, and since all the points are not equally important, that space is sampled by using a distribution that puts more weight on individuals with a higher fitness value. In order to achieve this, Metropolis-Hastings sampling is used, which is an extension of the Metropolis sampling towards non-symmetric stationary probability distributions [26].

3.3 Negative Slope Coefficient

The Negative Slope Coefficient is an algebraic measure for difficulty of the problem. It can be calculated in the following way: the fitness cloud C is divided into a certain number of segments $C_1, ..., C_m$ which are such that $\left(f_a, f_a^{'}\right)\epsilon C_j$ and $\left(f_b, f_b^{'}\right)\epsilon C_k$, where j ¡ k implies that $f_a < f_b$. An average fitness is calculated as:

$$\overline{f_i} = \frac{1}{C_i} \sum_{\left(f, f^{'}\right)\epsilon C_i} f \ . \tag{3}$$

and

$$\overline{f_i^{'}} = \frac{1}{C_i} \sum_{\left(f, f^{'}\right)\epsilon C_i} f^{'} \ . \tag{4}$$

The points $\left(\overline{f_i}, \overline{f_i^{'}}\right)$ can be viewed as polyline peaks, which successfully represent a 'skeleton' of the fitness cloud. For each of these segments a slope can be defined,

$$\overline{S_i} = \left(f_{i+1}^{'} - f_i^{'}\right) / \left(f_{i+1} - f_i\right) \ . \tag{5}$$

With this, an NSC is defined as:

$$nsc = \sum_{i=1}^{m-1} min\left(0, S_i\right) \ . \tag{6}$$

If NSC = 0 then the problem is easy, and if the NSC $<$ 0 then the problem is difficult and the NSC value shows to which extent it is difficult [19].

3.4 Unitation and Functions of Unitation

The Onemax, Onemix and Trap unitation functions are used [12] [20].

Definition 6 *Let s be a bit string of the length l. The unitation $u(s)$ of s is a function defined as:*

$$u(s) = u(s_i...s_l) = s_1 + ... + s_n = \sum_{i=1}^{l} s_i \; . \quad (7)$$

In other words, unitation represents the number of units in the bit string.

3.4.1 Onemax Function.

Onemax functions are generalizations of the unitation u(s), of a bit string s:

$$f(s) = du(s) \; . \quad (8)$$

where d in a general case is 1.

3.4.2 Trap Function.

Deb and Goldberg have defined the trap function as follows:

$$f(s) = \begin{cases} \frac{a}{z}(z - u(s)) & \text{if u(s)} \le \text{z} \\ \frac{b}{l-z}(u(s) - z) & \text{otherwise} \; . \end{cases} \quad (9)$$

Where 'a' represents a local optimum, 'b' is a global optimum and 'z' is a slope-change location. The trap function is completely deceptive if the following relationship is valid:

$$\frac{a}{b} = r \ge \frac{2 - \frac{1}{l-z}}{2 - \frac{1}{z}} \; . \quad (10)$$

This function is fully deceptive because a local optimum has an attractive basin that covers more space, and an isolated single spike representing global optimum which is the complement of the local optimum. Figure 1 shows a completely deceptive problem.

3.4.3 Onemix Function.

This function is a mixture of the Onemax problem and a Zeromax problem. Like these functions, it is a function of unitation u, which represents a number of 1's in a string. The new function becomes an Onemax function when the unitation values are higher than l/2. If the unitation values are lower, it becomes Onemax

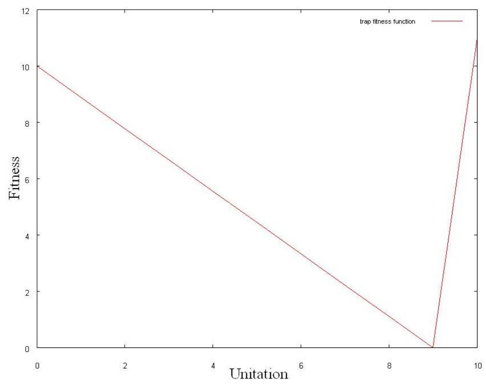

Figure 1: An example of Trap function.

when u is odd; otherwise it is a scaled version of Zeromax. Onemix is formally defined as:

$$f(s) = \begin{cases} (1+a)\left(\frac{l}{2} - u(s)\right) + \frac{l}{2} & \text{if g}(s) \\ u(s) & \text{otherwise} \; . \end{cases} \quad (11)$$

where g(s) is equal to 1 when u(s) is even and u(s)<1/2. Value 'a' represents a constant that is higher than 0 [20].

Figure 2 shows partialy deceptive problem.

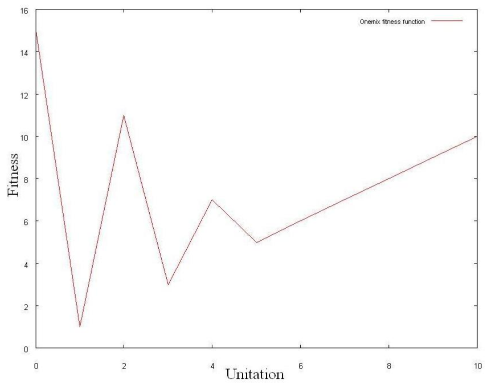

Figure 2: An example of Onemix function.

4 Experiments on Negative Slope Coefficient measure

Binary strings of length l = 10 were used in the experiments, by Metropolis-Hastings sampling of 100 individuals that constituted the first generation. These were selected and a standard bit-flip mutation was used with a p_m mutation coefficient for obtaining neighbors. For each individual, 10 neighbors were generated by the mutation operator and the one with the highest value of fitness was picked.

In the Onemax function example, all the experiments prove that the NSC work properly, which indicates that the Onemax is an easy problem to solve for a GA. This is in accordance with the performance measure. Values used for performance measure are displayed in the table below [19]. The performance measure indicates the fraction of runs in which the global optimum was found by generation 100.

Table 1: Performance Measure for Test Functions.

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

In order to present shortcomings of the NSC as a problem difficulty measure, a number of experiments with different parameters for the Trap Function were conducted. The following parameters for the Trap function were used; local optimum 'a' equals 10, global optimum 'b' equals 11 and slope-change location 'k' equals 9. The parameter predefined (marked predef. in the table) value set at yes marks that in this starting population each possible fitness value appears at least once. The parameters are similar to previous research [27], wherever this is possible. The division into segments in the first 4 experiments is made by setting the segment width to the constant value d, where d = 1, and in the other 5 experiments size driven bisection is used where 10 is the minimal number of points that may belong to a segment and 10% presents the minimal difference between the leftmost and the rightmost points contained in a segment. All results in tables represents the lowest values of NSC we calculated. Mean value is in more than half of experiments of value 0. The experiments show that the NSC is a good measure for the Trap function only when the mutation coefficient is relatively high, which can be seen in experiments 3, 4 and 5. Also, NSC appears to depend on members of first population (as displayed in experiment 3, table 3.). For the Onemix function, the NSC correctly presumes that the problem is difficult, but it incorrectly demonstrates that the Onemix problem is more difficult than the Trap problem, which does not correspond to the results achieved by the performance measure. The results obtained in the experiments are presented in tables below.

Table 2: NSC experimental results for Onemax function.

onemax function	pop_size	p_m	NSC
experiment 1	11	0.05	0
experiment 2	11	0.10	0
experiment 3	100	0.05	0
experiment 4	100	0.10	0

Table 3: NSC experimental for Trap function results for small population size and predefined individual values.

trap function	pop_size	p_m	predef.	NSC
experiment 1	11	0.05	yes	0
experiment 2	11	0.05	no	0
experiment 3	100	0.10	yes	-2
experiment 4	100	0.10	no	0

Table 4: NSC experimental results for Trap function.

trap function	pop_size	p_m	NSC
experiment 1	100	0.05	0
experiment 2	100	0.10	0
experiment 3	100	0.20	-0.79
experiment 4	100	0.30	-1.29
experiment 5	100	0.40	-0.025

Table 5: NSC experimental results for Onemix function.

onemix function	pop_size	p_m	NSC
experiment 1	11	0.05	0
experiment 2	11	0.10	-2
experiment 3	100	0.05	-4.335
experiment 4	100	0.10	-3.667
experiment 5	100	0.20	-0.596
experiment 6	100	0.30	-0.079
experiment 7	100	0.40	0

5 The New Negative Slope Coefficient Measure

In order to demonstrate whether a function is difficult for a GA or not, we propose a modification of the Negative Slope Coefficient, which is called new NSC. This measure is based on the relation between the number of units with value 1 on the x-line of the diagram and the fitness value for the individuals at the y-line of the diagram.

The experiments are based on the assumption that the maximum fitness value should be obtained. If there are no changes in the direction of the line and if the slope coefficient is positive, then the problem is easy: however if the slope coefficient is negative, then the problem is difficult. If there are changes to the slope segments of the line, then we calculate the slope for each of these segments. Each segment of the line S_1,...,S_m is defined minimally by two points with $V_1(x_1, y_1)$ and $V_2(x_2, y_2)$ coordinates. Then the slope of each segment S_i is defined by the formula:

$$a = \frac{y_2 - y_1}{x_2 - x_1} . \quad (12)$$

Values x_1 and x_2 represent two neighbor values of unitation and values y_1 and y_2 two values of fitness or fitness offspring.

The total slope for the fitness cloud with the number of units at the x-line and the individual fitness on the y-line is:

$$I_f = \sum_{i=1}^{m} a_i . \quad (13)$$

and for the fitness cloud with the number of units at the x-line and the neighbor fitness at the y-line:

$$\overline{I'_f} = \sum_{i=1}^{m} a_i . \quad (14)$$

A neighbor is chosen in the same way as in NSC measure where it represents an offspring with the maximum fitness value within 10 iterations. Finally, the new NSC amounts:

$$nsc = \left| \frac{I_f}{I'_f} \right| . \quad (15)$$

The higher the result, the harder the problem is for a GA to solve. The pseudo-code for the new NSC measure is:

```
function get_new_nsc (array_unitation)
{
 f[]= compute(array_unitation);
 fo[]= compute(array_unitation);
 i1= segments(array_unitation,f[]);
 i2= segments(array_unitation,fo[]);
 new_nsc=abs(i1/i2);
}
function segments(x,y)
{
 foreach value of x,y
 {
  array_x=slope change value(x);
  array_y=slope change value(y);
 }
 for i=2 to upper_bound(x)
 {
  y=array_y(i)-array_y(i-1);
  x=array_x(i)-array_x(i-1);
  sum=sum + y/x;
 }
 return sum;
}
```

Now the same experiments for the new NSC measures are repeated. When the new measure can not be calculated exactly, then words 'easy' and 'difficult' represent an indication of the problem hardness. The tables and figures below represent the obtained results.

Table 6: Experimental results for the new NSC measure for Onemax function.

onemax function	pop_size	p_m	new NSC
experiment 1	100	0.05	easy
experiment 2	100	0.10	easy
experiment 3	100	0.20	easy
experiment 4	100	0.30	easy

The figures that represent experiments on Onemax problems demonstrate that after mutation, the search space of possible solutions has become smaller, e.g. more individuals are closer to the global optimum.

Table 7: Experimental results for the new NSC measure for Trap function.

trap function	pop_size	p_m	new NSC
experiment 1	100	0.05	difficult
experiment 2	100	0.10	5.55
experiment 3	100	0.20	5.45
experiment 4	100	0.30	2.26

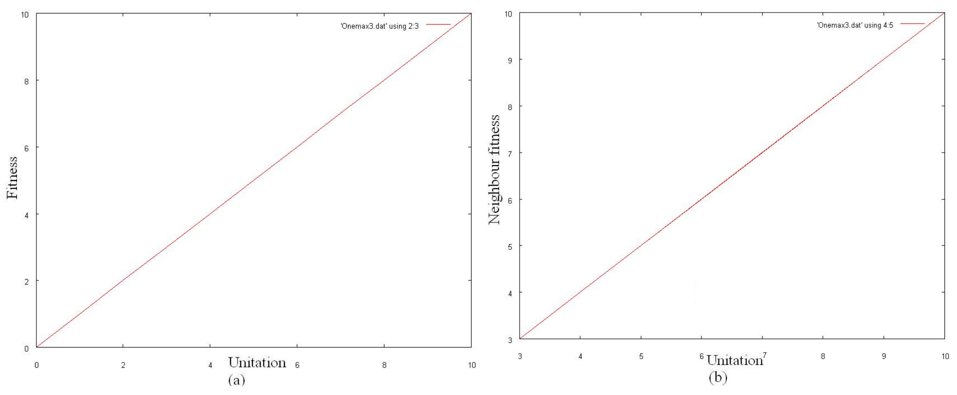

Figure 3: (a) graph unitation/fitness for Onemax function experiment 1, (b) graph unitation/fitness neighbour for Onemax function experiment 2.

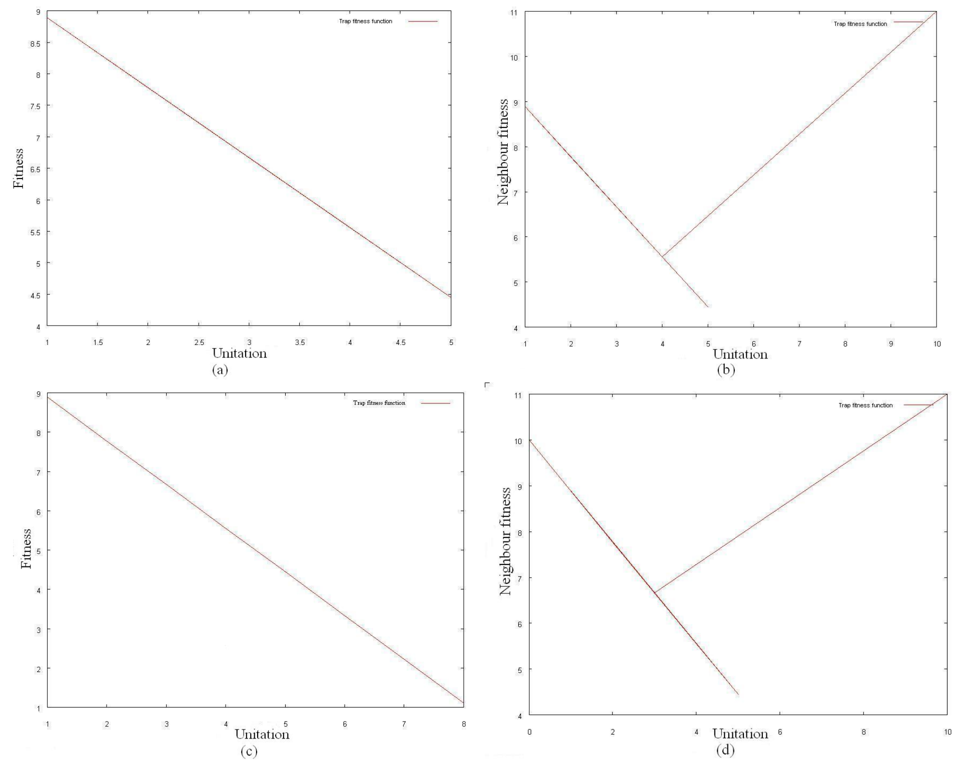

Figure 4: (a) graph unitation/fitness for Trap function experiment 2, (b) graph unitation/fitness neighbour for Trap function experiment 2, (c) graph unitation/fitness for Trap function experiment 4, (d) graph unitation/fitness neighbour for Trap function experiment 4.

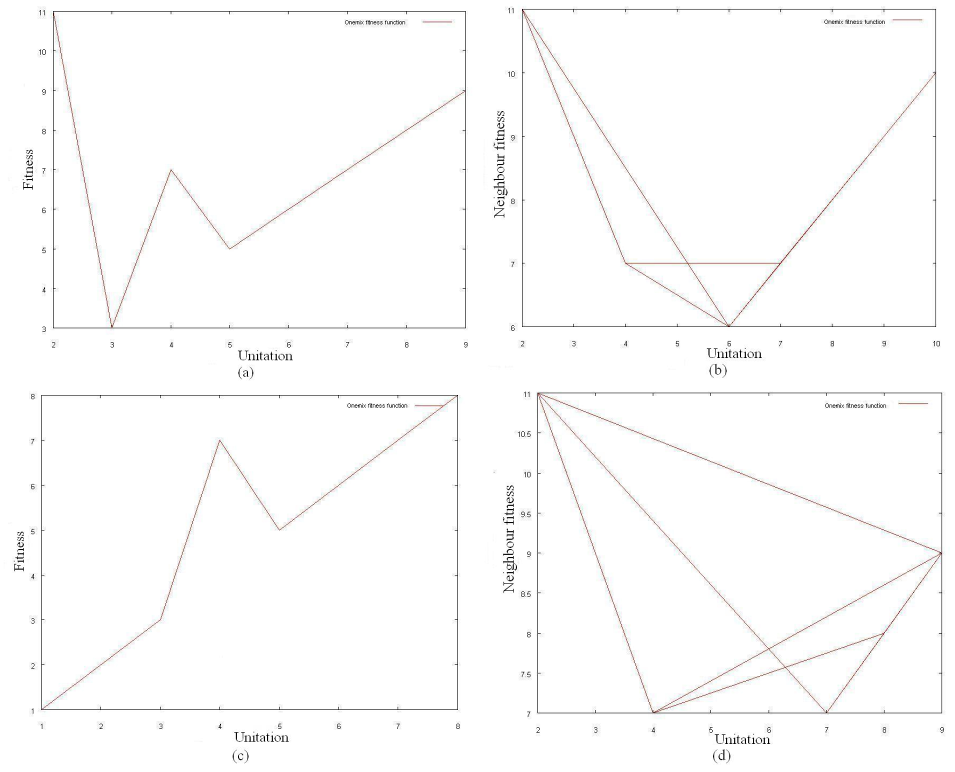

Figure 5: (a) graph unitation/fitness for Onemix function experiment 1, (b) graph unitation/fitness neighbour for Onemix function experiment 1, (c) graph unitation/fitness for Onemix function experiment 2, (d) graph unitation/fitness neighbour for Onemix function experiment 2.

Table 8: Experimental results for the new NSC measure for Onemix function.

onemix function	pop_size	p_m	new NSC
experiment 1	100	0.05	1.81
experiment 2	100	0.10	2.80
experiment 3	100	0.20	2.39
experiment 4	100	0.30	1.73

6 Conclusions and Future Work

Experiments on the original NSC have shown that it is a reliable difficulty measure only with relatively high mutation factors and that it is dependent on individuals of the first generation. Also, there is currently no formal underpinning for size driven bisection, so its parameters can only be selected in an arbitrary way. Experiments on the new NSC have shown that it also depends on individuals in first generation and on mutation rate. The new measure is, in the worst case, a reliable indicator of problem difficulty and at best a rather precise measure of problem hardness (difficult or easy). Another advantage of the new NSC is that there is no need to separate the points in segments via arbitrary segment size, size driven bisection or in any other way. The new NSC has some interesting features, but like the original NSC it will always be dependent of mutation rates and individuals of the first generation. Interesting results may be obtained by combining these two measures. Further experiments, on larger number of problem functions should be conducted in order to decide conclusively if it is possible to precisely calculate the difficulty of a problem. The first step must be to find completely reliable indicators of problem difficulty on larger number of different functions. It may also prove worthwhile to consider the problem difficulty from the level of bandit problems and other, more abstract levels of optimization. Deceptiveness seems to depict a relationship between optimization and problem instance, which may be more general than GAs alone.

References:

[1] Y. Borenstein, R. Poli: Information Landscapes and Problem Hardness. Genetic And Evolutionary Computation Conference, Proceedings of the 2005 conference on Genetic and evolutionary

computation, Washington DC, 2005, pp. 1425–1431

[2] K. Deb, D. E. Goldberg: Analyzing deception in trap functions. In: D. Whitley, (ed) Foundations of Genetic Algorithms 2, Morgan Kaufmann, San Franscisco, 1993, pp. 93-108

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

[4] T. Frankola, M. Golub, D. Jakobovi: Evolutionary Algorithms for the Resource Constrained Scheduling Problem, Proceedings of the 30th International Conference on Information Technology Interfaces, ITI 2008, Cavtat, Croatia, 2008, pp. 715–722.

[5] M. Golub: Genetski algoritam: dio I, in Croatian. Faculty of Electrical Engineering and Computing, University of Zagreb, 2004

[6] R. L. Haupt, S. E. Haupt: Practical Genetic Algorithms. John Wiley & Sons, 2004

[7] J. Horn, D. E. Goldberg: Genetic Algorithm Difficulty and the Modality of Fitness Landscapes. In: L. D. Whitley, and M. D. Vose, (eds), Proceedings of the Third Workshop on Foundations of Genetic Algorithms, Morgan Kaufmann, San Francisco, California, 1995, pp. 243–270

[8] T. Jones: Evolutionary algorithms, Fitness Landscapes and Search. Ph. D. thesis, The University of New Mexico, Albuquerque, New Mexico, 1995

[9] T. Jones, S. Forrest: Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In: L. J. Eshelman, (ed) Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kaufmann, San Franscisco, 1995, pp. 184–192

[10] M. Madras: Lectures on Monte Carlo Methods. American Mathematical Society, Providence, Rhode Island, 2002

[11] P. Makvandi, J. Jassbi, S. Khanmohammadi: Application of Genetic Algorithm and Neural Network in Forecasting with Good Data. Proceedings of the 6th WSEAS Int. Conf. on Neural Networks, Lisbon, Portugal, 2005, pp. 56–61

[12] O. J. Mengshoel, D. E. Goldberg, D. C. Wilkins: Deceptive and Other Functions of Unitation as Bayesian Networks. Symposium on Genetic Algorithms, Madison, Wisconsin, 1998

[13] B. L. Miller, D. E. Goldberg: Genetic Algorithms, Tournament Selection and the Efects of Noise. Technical Report, University of Illinois at Urbana-Champaign, 1995

[14] M. Mitchell: An Introduction to Genetic Algorithms. The MIT Press, Cambridge, 1999

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

[16] S. Picek: Decepcijski problemi, in Croatian. Faculty of Electrical Engineering and Computing, University of Zagreb, 2008

[17] S. Picek, M. Golub: The New Negative Slope Coefficient Measure, Proceedings of the 10th WSEAS International Conference on Evolutionary Computing, EC'09, 2009, Prag, Czech Republic, pp. 96–101

[18] K. P. Pieters: Effective Adaptive Plans. A Hypothetical Search Process. Advances in Systems, Computing Sciences and Software Engineering, Proceedings of SCSS05, Springer Netherlands, 2006, pp. 277–282

[19] R. Poli, L. Vanneschi: Fitness-Proportional Negative Slope Coefficient as a Hardness Measure for Genetic Algorithms. Genetic And Evolutionary Computation Conference, Proceedings of the 9th annual conference on Genetic and evolutionary computation, London, England, 2007, pp. 1345–1342

[20] R. Poli, A. H. Wright, N. F. McPhee, 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, Vancouver, Canada, 2006, pp. 395–402

[21] B. Rylander, J. Foster: Computational complexity and genetic algorithms. WSEAS International Conference on Evolutionary Computation, Tenerife Playa, Canary Islands, Spain 2001, pp. 6181–6185

[22] B. Rylander, J. Foster: Genetic Algorithms, and Hardness. WSEAS International Conference on Evolutionary Computation, Tenerife Playa, Canary Islands, Spain 2001, pp. 6431–6436

[23] M. Seyedkazemi: Designing Optimal PID controller with Genetic Algorithm In view of controller location in the plant. Proceedings of the 7th WSEAS International Conference on Signal Processing, Robotics and Automation, ISPRA '08, University of Cambridge, 2008, pp. 160–164

[24] M. Tomassini, L. Vanneschi, P. Collard, M. Clergue: A study of fitness distance correlation as a

difficulty measure in genetic programming. Evolutionary Computation,13 (2), MIT Press, Cambridge, 2005, pp. 213–239

[25] L. Vanneschi: Theory and Practice for Efficient Genetic Programming. Ph. D. thesis, Faculty of Science, University of Lausanne, Switzerland, 2004

[26] L. Vanneschi, M. Clergue, P. Collard, M. Tomassini, S. Vrel: Fitness clouds and problem hardness in genetic programming. In: K. D. et al.(ed) Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'04, LNCS vol. 3103, Springer, Berlin, Heidelberg, New York, 2004, pp. 690–701

[27] L. Vanneschi, M. Tomassini, P. Collard, S. Vrel: Negative slope coefficient. A measure to characterize genetic programming. In: P. Collet, M. Tomassini, M. Ebner, S. Gustafson, A. Ekrt (eds) Proceedings of the 9th European Conference on Genetic Programming, vol. 3905 of Lecture Notes in Computer Science, Springer, Budapest, Hungary, 2006 pp. 178–189

[28] T. Weise: Global Optimization Algorithms - Theory and Application, 2008

[29] L. D. Whitley: Fundamental Principles of Deception in Genetic Search. In: G. Rawlins, (ed) Foundations of Genetic Algorithms, Morgan Kaufmann, 1991, pp. 221–241

[30] S. Yang: Adaptive Group Mutation for Tackling Deception in Genetic Search. Digest of the Proceedings of the WSEAS Conferences, 2003