UNIVERSITY OF DORTMUND

REIHE COMPUTATIONAL INTELLIGENCE

COLLABORATIVE RESEARCH CENTER 531

Design and Management of Complex Technical Processes and Systems by means of Computational Intelligence Methods

On Classifications of Fitness Functions

Thomas Jansen

No. CI-76/99

Technical Report ISSN 1433-3325 November 1999

Secretary of the SFB 531 · University of Dortmund · Dept. of Computer Science/XI
44221 Dortmund · Germany

This work is a product of the Collaborative Research Center 531, "Computational Intelligence", at the University of Dortmund and was printed with financial support of the Deutsche Forschungsgemeinschaft.

On Classifications of Fitness Functions

Thomas Jansen
FB 4, LS 2, Univ. Dortmund, 44221 Dortmund, Germany
jansen@ls2.cs.uni-dortmund.de

Abstract

It is well-known that evolutionary algorithms succeed to optimize some functions efficiently and fail for others. Therefore, one would like to classify fitness functions as more or less hard to optimize for evolutionary algorithms. The aim of this paper is to clarify limitations and possibilities for classifications of fitness functions from a theoretical point of view. We distinguish two different types of classifications, descriptive and analytical ones. We shortly discuss three widely known approaches, namely the NK-model, epistasis variance, and fitness distance correlation. Furthermore, we consider another recent measure, bit-wise epistasis introduced by Fonlupt, Robilliard, and Preux (1998). We discuss shortcomings and counter-examples for all four measures and use this to motivate a discussion of possibilities and limitations of classifications of fitness functions in a broader context. and find out its shortcomings.

1 Introduction

Evolutionary algorithms are general search heuristics that can be used for global optimization. They are typically applied when there is not much knowledge about the objective function and there is no time for intense investigation of the problem. This is due to the circumstance that evolutionary algorithms are relatively simple to implement and that they are thought to be robust. Robustness means that evolutionary algorithms show above average performance on almost any objective function whereas more specialized optimization tools are superior on the class of objective functions they are designed for but inferior on the whole lot of the other functions. This popular point of view is visualized in a well-known figure in Goldberg's famous book on genetic algorithms (Goldberg 1989) and can be found implicitly and explicitly in many papers.

On the other hand, it is quite well-known, that averaged over *all* different fitness functions, all optimization algorithms (including a pure random walk) perform equal if one uses the number of different function evaluations as performance measure (Wolpert and Macready 1997). It is not useful to argue that taking only the different function evaluations into account is not appropriate for evolutionary algorithms. Using a dictionary, it can easily be achieved that no point in the search space is actually sampled twice. Furthermore, sampling any point more often than once obviously cannot improve the performance of an algorithm.

In spite of the correctness of this “no-free-lunch theorem“ (Wolpert and Macready 1997) the result is not too interesting. It is easy to see, that averaging over all different fitness functions does not match the situation of black-box optimization in practice. It can even be shown that in more realistic optimization scenarios there can be no such thing as a no-free-lunch theorem (Droste, Jansen, and Wegener 1999).

We conclude that the classification of objective functions in order to determine whether they can be successfully optimized by (certain) evolutionary algorithms is both, practically relevant and theoretically justified. Before objective functions are optimized by evolutionary algorithms there usually is some mapping to a fitness function. We omit this step that is subject to research of itself and directly consider the optimization of fitness functions. In order to simplify the task we restrict ourselves to fitness functions $f : \{0,1\}^n \to \mathbb{R}$. We know that the optimization of functions $g : \mathbb{R}^n \to \mathbb{R}$ may lead to different techniques. Already the analysis of evolutionary algorithms heavily depends on the chosen representation, whether it is discrete or continuous, see e. g. Rudolph's analyses of evolution strategies (Rudolph 1997). But in concrete implementations on digital computers real numbers are (in most cases implicitly) mapped to bit strings. Thus, we do not consider our simplification to be a fundamental restriction.

We distinguish two different types of classifications, namely descriptive and analytical ones. A descriptive classification defines a class of fitness functions with some common property. It can be considered to be helpful in our context if this property can be related to the difficulty of optimizing these functions. Note, that it may be a hard problem to decide whether a given fitness function belongs to the defined class.

An analytical classification is a kind of algorithm that takes a fitness function as input and yields some kind of classifying attribute as output. Typically, the output is some number, but more complex attributes may and are also be used.

Maybe the best kind of analytical classification one could like to have is the following. Assume we have a set of optimization algorithms $\{A_1, \ldots, A_n\}$. Then we look for a classification algorithm that takes a fitness function as input and computes the index i of the optimal algorithm A_i for f. By applying first the classification algorithm and then the optimization algorithm A_i we wand to reduce the time needed to optimize f. The no-free-lunch theorem rules out that such a classification is possible for all fitness functions. In fact, the best that can be achieved is an improvement on a proper subclass of all functions. Thus, it is a necessary element of such an analytical classification, that the class of functions it works well for is defined. We see the need for a combination of analytical and descriptive classifications here.

The problem of classification of fitness functions has already been subject to intense research, probably most often in the context of GA-hardness. Among the most prominent hardness measures are epistasis variance (Davidor 1991) and fitness distance correlation (Jones and Forrest 1995). An overview is given by Naudts (1998). Another approach can be seen in the NK-model (Kauffman 1993), though introduced in another context and for other reasons. We cannot hope to solve the multitude of open problems that are still unsolved. We thus intend to clarify and enrich the field by providing another perspective of the problem.

In Section 2 we describe three well-known classifications and one more recent one. In Subsection 2.1 we discuss the NK-model, a descriptive classification, which is of special interest to us. We give a very brief overview of epistasis variance and fitness distance correlation in Subsection 2.2 and Subsection 2.3, respectively. Both are analytical classifications. In Subsection 2.4 we consider another quite recent approach to the classification of fitness functions, namely bit-wise epistasis (Fonlupt, Robilliard, and Preux 1998), which is an analytical classification, too. We present arguments why bit-wise epistasis is not an appropriate measurement for the hardness of functions with respect to function optimization by means of evolutionary algorithms. Then we try to gain a broader perspective and look for fundamental restrictions and possible directions of research in the field of fitness function classification in Section 3. Finally, Section 4 summarizes and gives perspectives.

2 Four Different Classifications

In this section we discuss four different classifications, one descriptive and three analytical ones. We give serious reasons why the classifications are of limited use, only, and may be considered as not too helpful in practice. This section is intended to clarify the common problems of classifications and leads the way to a more constructive discussion in the following section.

2.1 The NK-model

As a first example of a classification of fitness functions we consider the NK-model introduced by Kauffman (1993) (see Altenberg (1997b) for an overview). Originally, it defines a kind of probabilistic class of fitness functions from $\{0,1\}^N$ (where N is n in our notation) to $[0,1]$, with tuneably "ruggedness" or "epistasis", i. e. interaction between different bits. We present a version of the NK-model (and turn to the notation nk-model), that fits better within the context of classification.

Definition 1. *A fitness function $f\colon \{0,1\}^n \to \mathbb{R}$ belongs to the class of nk* functions*, where $k \in \{0,\dots,n-1\}$, if there are n functions $g_1\colon \{0,1\}^{k+1} \to \mathbb{R}$, ..., $g_n\colon \{0,1\}^{k+1} \to \mathbb{R}$, and n sets $I_1 = \{i_{1,1},\dots,i_{1,k}\}$, ..., $I_n = \{i_{n,1},\dots,i_{n,k}\}$, such that*

$$f(x) = g_1\left(x_1, x_{i_{1,1}}, \dots, x_{i_{1,k}}\right) + \cdots + g_n\left(x_n, x_{i_{n,1}}, \dots, x_{i_{n,k}}\right)$$

for all $x = x_1x_2\cdots x_n \in \{0,1\}^n$. The fitness function f is called adjacent nk function*, if additionally $I_1 = \{2,3,\dots,1+k \bmod n\}$, ..., $I_n = \{1,2,\dots,k\}$ holds.*

Obviously, the nk-model defines a partition of the class of all fitness functions into sets S_k of functions that are nk functions but not $n(k-1)$ functions. The idea is, that the parameter k defines the degree of interaction between different bits that is allowed. For $k=0$ no interaction is permitted. Such functions are called bit-wise separable (or linear) and can be optimized easily by many algorithms (see e. g. Droste, Jansen, and Wegener (1998c) for the expected running time of a simple evolutionary algorithm on the class of linear functions). Furthermore, for $k = n-1$ difficult functions like the class of "needle in a haystack" functions (that is $\{f_a | a \in \{0,1\}^n\}$ with $f_a(a) = 1$ and $f_a(x) = 0$ for $x \neq a$) are included, that require exponential computational effort for all optimization algorithms.

But the classes with intermediate k cause some problems. For the moment, we leave the field of evolutionary algorithms and take a look at the nk-model from a complexity theoretical point of view. Then we look for upper resp. lower bounds on the computational effort that is sufficient resp. needed to optimize a nk function. It is easy to see that even for $k=2$ optimizing such a function is NP-hard (Thompson and Wright 1996). In fact, it can be reduced to the NP-hard MAX-2-SAT problem (for an introduction see Garey and Johnson (1979)). For $k=1$ a polynomial algorithm can be found (Thompson and Wright 1996). Things change dramatically, if we consider adjacent nk functions. In this case, using a dynamic programming approach (see Cormen, Leiserson, and Rivest (1990) for an introduction) leads to an optimization algorithm with running time $\mathcal{O}\left(2^k n\right)$, which is polynomial for $k = \mathcal{O}(\log n)$ (Weinberger 1996).

For evolutionary algorithms no results are proven for general k. Experiments show little difference for the two cases of the adjacent nk functions as well as the ones with arbitrary neighborhood (Kauffman 1993). Weinberger (1996) computes the correlation between pairs of fitness in the two different versions of the nk-model and finds only little difference, too.

As general nk functions are NP-hard for $k \geq 2$, there is no hope that they are appropriate for partitioning the class of all fitness functions according to their difficulty for evolutionary algorithms. For adjacent nk functions things may be different. But as empirical experiences suggest that for evolutionary algorithms there is little difference between the two models, we may speculate that adjacent nk-models do not deliver a useful partition, either. Furthermore, a recent paper of Naudts and Kallel (1999) demonstrates in an empirical way that already the class of adjacent $n1$ functions contains functions that can be of extremely different difficulty to a (standard) genetic algorithm.

2.2 Epistasis variance

Epistasis variance is introduced by Davidor (1991) as a simple statistic for measuring the hardness of a function. It gives a measure of the amount of nonlinearity that can be found. It is formally given by

$$\varepsilon = \sqrt{2^{-n} \sum_{x \in \{0,1\}^n} \left(f(x) - 2^{1-n} \sum_{i=1}^{n} \sum_{\substack{y \in \{0,1\}^n \\ y_i = x_i}} f(y) + \frac{n-1}{2^n} \sum_{y \in \{0,1\}^n} f(y) \right)^2}$$

(Naudts, Suys, and Verschoren 1997) and is obviously an analytical classification. Note, that the computation of the exact value takes $\Omega\left(2^n\right)$ steps in general. Naudts, Suys, and Verschoren define a normalized epistasis value and remark that the maximal value is obtained by so called "camel functions". These functions are constant for all points in the search space except for one point s and its bitwise complement $\overline{s}$, which are the global maxima. Obviously, these functions are almost as hard to optimize as the "needle in a haystack" functions discussed above. Let as consider the instance of CAMEL with $s = \mathbf{0}$, the all zero bit string (so the complement is $\mathbf{1}$, the all one string), and the two different function values 0 and 1. Then we have

$$\sum_{y \in \{0,1\}^n} \textsc{Camel}(y) = 2,$$

and for all $x \in \{0,1\}^n$

$$\sum_{\substack{y \in \{0,1\}^n \\ y_i = x_i}} \textsc{Camel}(y) = \begin{cases} 1 & \text{if } x \in \{\mathbf{0}, \mathbf{1}\}, \\ 2 & \text{otherwise.} \end{cases}$$

This leads to

$$
\begin{aligned}
\varepsilon_{\textsc{Camel}} &= \left(\left(2\left(1-\left(n\left(\frac{1}{2^{n-1}}-\frac{1}{2^{n-1}}\right)+\frac{1}{2^{n-1}}\right)\right)^2\right.\right.\\
&\qquad \left.\left. +\,(2^n-2)\left(0-\left(n\left(\frac{2}{2^{n-1}}-\frac{1}{2^{n-1}}\right)+\frac{1}{2^{n-1}}\right)\right)^2\right)2^{-n}\right)^{1/2}\\
&= \sqrt{\left(2\left(1-\frac{1}{2^{n-1}}\right)^2+(2^n-2)\left(-\frac{n+1}{2^{n-1}}\right)^2\right)2^{-n}}
\end{aligned}
$$

for the epistasis variance of Camel. We do not care about the size of this value here, but compare it with the epistasis variance of another function denoted as $\overline{\textsc{Camel}}$. In particular consider the instance of $\overline{\textsc{Camel}}$ with $\overline{\textsc{Camel}}(\mathbf{0}) = \overline{\textsc{Camel}}(\mathbf{1}) = 0$ and $\overline{\textsc{Camel}}(x) = 1$ for $x \notin \{\mathbf{0}, \mathbf{1}\}$. Obviously, $\overline{\textsc{Camel}}$ is extremely simple to maximize for any reasonable algorithm. But we have

$$
\sum_{y\in\{0,1\}^n} \overline{\textsc{Camel}}(y) = 2^n - 2
$$

and for all $x \in \{0,1\}^n$

$$
\sum_{\substack{y\in\{0,1\}^n\\ y_i=x_i}} \overline{\textsc{Camel}}(y) = \begin{cases} 2^{n-1}-1 & \text{if } x \in \{\mathbf{0},\mathbf{1}\},\\ 2^{n-1}-2 & \text{otherwise,}\end{cases}
$$

leading to

$$
\begin{aligned}
\varepsilon_{\overline{\textsc{Camel}}} &= \left(\left(2\left(0-\left(n\left(1-\frac{1}{2^{n-1}}-1+\frac{1}{2^{n-1}}\right)+1-\frac{1}{2^{n-1}}\right)\right)^2\right.\right.\\
&\qquad +\,(2^n-2)\left(1-\left(n\left(1-\frac{2}{2^{n-1}}-1+\frac{1}{2^{n-1}}\right)\right.\right.\\
&\qquad\quad \left.\left.\left.\left. +1-\frac{1}{2^{n-1}}\right)\right)^2\right)2^{-n}\right)^{1/2}\\
&= \sqrt{\left(2\left(1-\frac{1}{2^{n-1}}\right)^2+(2^n-2)\left(\frac{n+1}{2^{n-1}}\right)^2\right)2^{-n}}
\end{aligned}
$$

We conclude, that epistasis variance may associate exactly the same value to extremely simple and extremely difficult functions.

2.3 Fitness distance correlation

Fitness distance correlation is introduced by Jones and Forrest (1995) as an alternative analytical measure of search difficulty. It is based on the idea of a "fitness landscape" where the points in the search space together with their fitness form a kind of landscape in the following sense: there is a weighted neighborhood relationship defined in the search space and we interpret each point's fitness as its height (Jones 1995). We assume that we are maximizing, so one is looking for a point with maximal height. To establish a meaningful relationship between the optimization problem at hand and the fitness landscape, it is necessary that the neighborhood relationship depends on the algorithm used. However, often simply Hamming distance is employed for the definition of the fitness landscape.

Fitness distance correlation measures the extent to which high fitness values correlate with small distance to a global optimum. If $F = \{f_1, \ldots, f_k\}$ is the set of fitness values and $D = \{d_1, \ldots, d_k\}$ the corresponding distances to the nearest global optimum, let $\overline{f}$ denote the mean of F, $\overline{d}$ the mean of D over all $x \in \{0,1\}^n$. Let s_F and s_D denote the standard deviation of F and D. Let c_{FD} with

$$c_{\mathrm{FD}} = k^{-1} \sum_{i=1}^{k} \left(f_i - \overline{f}\right)\left(d_i - \overline{d}\right)$$

be the covariance of F and D. Then, the fitness distance correlation r is formally defined as

$$r = \frac{c_{\mathrm{FD}}}{s_F \cdot s_D}.$$

Again, the computational effort for the computation of the exact value is $\Omega\left(2^n\right)$. Though it is reported by Jones and Forrest (1995) that fitness distance correlation correctly classifies the hardness of many functions, it is well-known that there are counterexamples. Altenberg (1997a) constructs a function that is easy to optimize but shows no relationship between Hamming distance from the global optimum and fitness. Quick, Rayward-Smith, and Smith (1998) present a class of functions with the same property and the advantage to be well-structured and thus understandable. Formally, the ridge function is defined by

$$\textsc{Ridge}(x) = \begin{cases} n + 1 + ||x||_1 & \text{if } \exists i \in \{0, 1, \ldots, n\} : x = 1^i 0^{n-i}, \\ n - ||x||_1 & \text{otherwise,} \end{cases}$$

where $||x||_1$ denotes the number of ones in x and $1^i 0^{n-i}$ is the bit string of length n with first i consecutive ones followed by $n-i$ consecutive zeros.

The global optimum of Ridge is $\mathbf{1}$, the all one string, where we have $\text{Ridge}(\mathbf{1}) = 2n + 1$. But for all 2^n string except for the $n + 1$ strings of the form 1^i0^{n-i} higher fitness values are correlated with fewer ones in the string. Thus, as fitness distance correlation averages over all 2^n strings, the correlation is very small indicating a very difficult function. Nevertheless, Ridge is fairly easy to optimize. We consider a simple hill climber to see that. Obviously, the all zero string $\mathbf{0}$ is easy to find. Then, a path of length n that can be easily followed leads to the global optimum: there is always exactly one string with Hamming distance 1 to the current string x that has better function value than x. Any reasonable hill climber finds this string in $\mathcal{O}(n)$ steps. So, after $\mathcal{O}(n^2)$ steps the global optimum is reached.

2.4 Bit-wise epistasis

Another quite recent attempt of a classification of fitness functions is the introduction of bit-wise epistasis as measurement of the problem difficulty by Fonlupt, Robilliard, and Preux (1998). The basis of this measurement is the epistasis, i. e. the interaction between different bits, like for nk-models. Like epistasis it is an analytical classification.

Definition 2. *The bit-wise epistasis is defined for a bit i, where $1 \leq i \leq n$, and a fitness function $f : \{0,1\}^n \to \mathbb{R}$. Let $\Sigma_i \subseteq \{0,1,*\}^n$ be the set of all schemata, such that for all $x = b_1b_2\cdots b_n \in \Sigma_i$ we have $b_i = *$ and $b_j \in \{0,1\}$ for $j \neq i$. Obviously, we have $|\Sigma_i| = 2^{n-1}$. For $x \in \Sigma_i$ we define $x_0 \in \{0,1\}^n$ as the bit string that is obtained by replacing the "$*$" at the i-th position in x by a 0. Analogously x_1 is defined as x with a 1 at the i-th position. Then,* bit-wise epistasis σ_i^2 *for bit i is given by*

$$\sigma_i^2 := 2^{1-n} \sum_{x\in\Sigma_i} \left(2^{1-n} \left(\sum_{y\in\Sigma_i} f(y_0) - f(y_1) \right) - (f(x_0) - f(x_1)) \right)^2 .$$

We present two classes of functions and one special function that we use to exemplify the inherent problems of this new difficulty measure. The first class of functions $\{f_a : \{0,1\}^n \to \mathbb{R} \mid a \in \{0,1\}^n\}$ consists of the already mentioned "needle in a haystack" functions with $f_a(a) = 1$ and $f_a(x) = 0$ for $x \neq a$. The second class of functions $\{g_k : \{0,1\}^n \to \mathbb{R} \mid k \in \mathbb{N}\}$ is defined by $g_k(x) := k \cdot \prod_{i=1}^n x_i$ and is simply a variant of $f_{11\cdots1}$. Finally, the function $\text{LeadingOnes} : \{0,1\}^n \to \mathbb{R}$ is defined by

$$\text{LeadingOnes}(x) := \sum_{i=1}^{n} \prod_{j=1}^{i} x_j$$

and simply counts the number of leading ones in x.

The computation of the precise bit-wise epistasis for an objective function f is extremely computationally expensive, in general. All 2^n function values of f have to be known for that. It follows, that in practice one has to use an estimation of the bit-wise epistasis that is based on a polynomially bounded number of function evaluations. Before we discuss this difficulty in some more detail we consider the bit-wise epistasis of f_a to demonstrate that even the knowledge of the exact bit-wise epistasis is not necessarily too helpful when one wants to determine a global optimum.

Due to the definition of f_a the exact values of the bit-wise epistasis can be easily determined. For all i with $1 \leq i \leq n$ we have

$$\sigma_i^2 = \frac{1}{2^{n-1}} \sum_{x \in \Sigma_i} \left(f_a\left(x_1\right) - \frac{1}{2^{n-1}} \right)^2 = \frac{1}{2^{n-1}} - \frac{1}{2^{2n-2}}.$$

We recognize, that the values σ_i^2 are equal for all i and do not depend on a so that no information about the global optimum a can be won from the exact values of the bit-wise epistasis.

For g_k the bit-wise epistasis σ_i^2 can be determined analogously. We have

$$\sigma_i^2 = \frac{1}{2^{n-1}} \sum_{x \in \Sigma_i} \left(g_k\left(x_1\right) - \frac{1}{2^{n-1}} \right)^2 = \frac{(2^{n-1}-1) \cdot k^2}{2^{3n-3}} + \frac{1 - 2^n + 2^{2n-2}}{2^{2n-2}}.$$

Obviously, all σ_i^2 grow with k. Since k may take any value in $\mathbb{N}$, the bit-wise epistasis may grow arbitrarily large.

For practical purposes the bit-wise epistasis is computed by choosing for each position i a polynomial number $p(n)$ of schemata from Σ_i. There is only one schema $x \in \Sigma_i$ that yields the all one string for x_1. Therefore, with probability $\rho = (1 - 2^{-(n-1)})^{n \cdot p(n)}$ the estimated bit-wise epistasis is 0 for each i. Obviously, we have $\lim_{n \to \infty} \rho = 1$, so with g_k we have functions for which almost surely the difference between the estimated bit-wise epistasis and the exact values can be arbitrarily large.

The functions f_a and g_k have in common that they are all very difficult to optimize. The expected number of function evaluations before the optimum is found is exponential for f_a; the same holds for a generalized version of g_k where the bit string with optimal function value k is not the same for all functions. So, one may want to argue that for more practical, i. e. less hard functions the bit-wise epistasis may well be a valuable measure of problem difficulty and—what is more important—may be a valuable tool to speed up optimization. Fonlupt, Robilliard, and Preux (1998) suggest that bits with higher bit-wise epistasis are mutated with a higher probability than bits with lower bit-wise epistasis.

We consider the bit-wise epistasis of LeadingOnes, which is not hard to compute, to investigate this idea. We consider σ_i^2 and distinguish two cases concerning the bit strings in $x \in \Sigma_i$. If there is at least one zero in x, let j be the position of the leftmost zero in x. If $j < i$ holds, we have $\text{LeadingOnes}(x_0) - \text{LeadingOnes}(x_1) = 0$. If $j > i$ holds, we have $\text{LeadingOnes}(x_0) - \text{LeadingOnes}(x_1) = (i-1)-(j-1) = i-j$. In the second case, x does not contain any zero at all and we have $\text{LeadingOnes}(x_0) - \text{LeadingOnes}(x_1) = (i-1)-n$. This leads us directly to

$$\begin{aligned}\sigma_i^2 &= \frac{1}{2^{n-1}} \sum_{x \in \Sigma_i} \left(\frac{1}{2^{n-1}} \left(i-1-n+\sum_{j=i+1}^{n} 2^{n-j}(i-j) \right) \right. \\ &\qquad \left. - \left(\text{LeadingOnes}(x_0) - \text{LeadingOnes}(x_1)\right) \right)^2 \\ &= 2^{1-n} \left(\left(2^{1-n} \left(i-1-n+\sum_{j=i+1}^{n} 2^{n-j}(i-j) \right) - (i-1-n) \right)^2 \right. \\ &\qquad \left. + \sum_{k=i+1}^{n} 2^{n-k} \left(2^{1-n} \left(i-1-n+\sum_{j=i+1}^{n} 2^{n-j}(i-j) \right) - i+k \right)^2 \right).\end{aligned}$$

For $i = n$ we get

$$\sigma_n^2 = 2^{-n+1}\left(2^{-n+1}-1\right)^2 = \mathcal{O}\left(2^{-n}\right).$$

For $i = 1$ a lengthy by straightforward calculation shows that

$$\sigma_1^2 = 2 + 2^{-n+1} - n2^{-n+2} - 2^{-2n+2} = \Omega(1)$$

holds. Fonlupt, Robilliard, and Preux (1998) suggest that the mutation probabilities of the single bits are chosen proportionally to the bit-wise epistasis. For LeadingOnes this implies that the last bit gets an exponentially small mutation probability, immediately implying exponentially large expected running times. We like to mention that LeadingOnes is in fact quite easy to optimize. Rudolph (1997) proves that the well-known $(1+1)$ evolutionary algorithm optimizes it in expected time $\mathcal{O}\left(n^2\right)$; indeed the expected running time of this simple EA is $\Theta\left(n^2\right)$ (Droste, Jansen, and Wegener 1998b). But even increasing the mutation probability for the leading bits in a less dramatic way is apparently not too clever an idea. It implies that the probability for a successful mutation is decreased so that overall the expected running time is increased.

We see that bit-wise epistasis is a measurement for the difficulty of objective functions that is at least problematic. Namely we proved

(1) that knowing the exact values of the bit-wise epistasis may be of no help at all for finding an optimum,

(2) that the estimates that have to be used in practice may be arbitrarily bad with high probability, and

(3) that even an heuristic use of bit-wise epistasis for adjusting mutation probabilities may be of no help.

It is not excluded that in practice bit-wise epistasis may turn out to be a helpful tool in some cases. But it is definitely not the ultimate answer to the question how the difficulty of objective functions can be measured and more sound arguments in favor of its usefulness have still to be found.

3 Limits and Perspectives of Fitness Function Classification

The previous section, in particular the work of Fonlupt, Robilliard, and Preux (1998) can be seen as a motivation for this section. They present a new way to measure the difficulty of a function with respect to optimization by evolutionary algorithms. As we have shown in the previous section, there are serious reasons to doubt the usefulness of this measure. The same holds for the other measures considered. They all may deliver wrong predictions. Obviously, it is not sensible to continue presenting new ways of function classifications while leaving the task of proving that the presented classifications are inappropriate in some way to other researchers. Therefore, we try to clarify the task of classifying functions and gain a hopefully helpful perspective from a theoretical point of view.

In the context of optimization with evolutionary algorithms usually the underlying optimization scenario is that of black-box optimization. It therefore seems reasonable to perform the task of classification of fitness functions in a black-box scenario, too. In this case there is no hope to find a measure, that allows us to distinguish between very simple and very difficult functions with polynomially bounded computational effort and acceptable probability in all cases. In order to distinguish between a constant function and a "needle in a haystack" function f_a one has to sample the point a. But, if, for a polynomial p, at most $p(n)$ points in the search space are sampled, the probability of finding a is

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

so with probability $1 - 2^{-\Omega(n)}$ we fail to distinguish f_a, which is hard to optimize, from a trivially optimized constant function.

As Heckendorn and Whitley (1999) point out, things may change, if we assume that the fitness function is given as a mathematical expression, e. g., a polynomial. But it is obvious that the optimization of polynomials of degree 2 is NP-hard (as MAX-2-SAT can be coded that way). Furthermore, for a simple EA, Droste, Jansen, and Wegener (1998a) prove in a more direct way that already polynomials of degree 2 can be very hard to optimize. Therefore, the question remains unanswered how from the representation of a fitness function as mathematical expression a useful classification can be extracted.

One may ask whether concentrating on concrete, maybe simplified, algorithms enables us to find a helpful classification of the set of all fitness functions with respect to this chosen algorithm. Though this cannot in general be ruled out, there are well-known examples that make a success of this approach for interesting algorithms unlikely. Local search is a well-known, simple general search heuristic, that is e. g. employed in simulated annealing (Kirkpatrick, Gelatt, and Vecchi 1983). We consider only such local search heuristics and optimization problems, that it is easy to decide whether a local optimum is reached. It is a natural question to ask *which* local optimum will be reached by the considered local search heuristic if started in some given point in the search space. It is known that answering this question is NP-hard (Johnson, Papadimitriou, and Yannakakis 1988), in fact it is PSPACE-complete (Papadimitriou, Schäffer, and Yannakakis 1990). We see, that even for local search deciding whether the global optimum is reached at all can be PSPACE-complete. Doing this for some evolutionary algorithm is probably too ambitious.

We recognize that solving the problem of classifying fitness functions in a black-box scenario is not possible when restricted to measurements with polynomially bounded computational effort. Furthermore, there are good reasons to believe that restricting the interest to some specific evolutionary algorithm does not simplify this task significantly. So, we are left with the question what can be achieved at all. We try to characterize three different ways, where fruitful research seems possible.

We know that in practice it is not possible to distinguish between a constant and a "needle in a haystack" function. This is due to their extreme "closeness", i. e. the two functions differ at only one point in the search space. If two functions differ at substantially more points, then there is a not too small probability to distinguish between them by random sampling. This approach has similarities to learning theory, we refer to an article on property testing by Goldreich, Goldwasser, and Ron (1998) for an introduction and interesting examples. It is an important open question how property testing can be related to the classification of fitness functions. What is needed are sets of fitness functions, such that two functions of different sets are in a well-defined sense far from each other, and that all functions of one set are of similar difficulty with respect to optimization by evolutionary algorithms. An interesting approach that can be seen as a kind of classification in this sense is presented by Garnier and Kallel (1999). There, estimates are given for the number of random starting points for local search that are needed in order to find all local maxima of a given function. For certain classes of functions tight estimates are derived. But furthermore, the practically more relevant inverse situation is investigated, too. For an unknown function the number of local optima is estimated by the application of random sampling, local search, and the confidence level is estimated using a statistical test. We remark that the problem of guaranteeing a reasonable running time is left open as the running time of the local search is not bounded.

Another approach is the identification of more or less natural classes of fitness functions that have similar difficulty. This means constructing a purely descriptive classification. If functions from such a class can be expected to occur in practice, this approach leads to valuable knowledge. We mention linear functions as a successful example (Droste, Jansen, and Wegener 1998c). As a negative example we mention unimodal functions, that can be of highly differing difficulty (Droste, Jansen, and Wegener 1998b).

The third possible way in which one may hope to gain a deeper understanding of evolutionary algorithms is the perhaps least ambitious one. It is the analysis of concrete, maybe simplified, evolutionary algorithms on concrete fitness functions. There are many examples for this kind of research, we name (Droste, Jansen, and Wegener 1998c; Garnier, Kallel, and Schoenauer 1999; Rudolph 1997) as examples. Though it may seem that such an approach is quite far away from the classification of fitness functions, this is not necessarily the case. An in-depth analysis of a concrete evolutionary algorithm provides us with insights in the principle working of the different search operators evolutionary algorithms employ. Finding good examples, functions that paradigmatically show successes and failures of an EA, can enable us to formulate more general circumstances under which such algorithms will succeed or fail. Proven results may serve as a more solid ground for an understanding of when and why evolutionary algorithms should be applied. And this is what is the aim of the classification of fitness functions, after all.

4 Conclusions

It is clear by now that there are serious limitations to what can be achieved by a classification of fitness function. An efficient, overall improving and perfect classification scheme cannot exist. Nevertheless, there are partial successes in the field and we speculate that more is possible.

We summarize what we learned by reviewing four existing classifications. There are two different ways to approach the problem of classification of fitness functions, a descriptive and an analytical one. Though there are fundamental differences between the two approaches, there are strong connections between them. A descriptive classification is a valuable piece of knowledge, since it tells us what functions share common properties that are relevant to the task of optimization. In order to become practically relevant, a method is needed that allows us to find out whether a given function belongs to the defined class. We see that a kind of analytical classification is needed, that allows to efficiently determine whether a given function belongs to a specific class.

On the other hand, an analytical classification alone cannot be a useful tool in all situations. Either it needs exponential running time to classify some functions or wrong predictions will be drawn. In Section 2 we discussed examples that prove that all the analytical classifications we considered can lead to wrong predictions even when given exponential time. Of course, all example functions used are purely artificial and may be seen as “pathological cases”. But as long as an analytical classification is not accompanied by a definition of the classes of functions, i. e., a descriptive classification, such “pathological” examples are allowed.

Acknowledgements

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the Collaborative Research Center “Computational Intelligence” (531). It was written as a contribution to the second Evonet summer school on theoretical aspects of evolutionary computing (Antwerp, September 1999). The author wishes to thank Leila Kallel, Bart Naudts, Karsten Tinnefeld, and Ingo Wegener for helpful comments and discussions.

References

L. Altenberg (1997a). Fitness distance correlation analysis: An instructive counter-example. In T. Bäck (Ed.), *Proceedings of the 7th International*

Conference on Genetic Algorithms (ICGA '97), 57–64. Morgan Kaufman, San Francisco, CA.

L. Altenberg (1997b). NK fitness landscapes. In *Handbook of Evolutionary Computation*, B2.7.2. Oxford University Press, Oxford.

T. H. Cormen, C. E. Leiserson, and R. L. Rivest (1990). *Introduction to Algorithms*. McGraw-Hill, New York, NY.

Y. Davidor (1991). Epistasis variance: A viewpoint on GA-hardness. In G. J. E. Rawlins (Ed.), *Proceedings of the 1st Workship on Foundations of genetic algorithms (FOGA)*, 23–35. Morgan Kaufman, San Mateo, CA.

S. Droste, T. Jansen, and I. Wegener (1998a). On the analysis of the $(1+1)$ evolutionary algorithm. Technical Report CI-21/98, University of Dortmund, SFB 531, Dortmund, Germany.

S. Droste, T. Jansen, and I. Wegener (1998b). On the optimization of unimodal functions with the (1+1) evolutionary algorithm. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel (Eds.), *Parallel Problem Solving from Nature (PPSN V)*, 47–56. Springer, Berlin, Germany.

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

S. Droste, T. Jansen, and I. Wegener (1999). Perhaps not a free lunch but at least a free appetizer. In *Genetic and Evolutionary Computation Conference (GECCO '99)*.

C. Fonlupt, D. Robilliard, and P. Preux (1998). A bit-wise epistasis measure for binary search space. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel (Eds.), *Parallel Problem Solving from Nature (PPSN V)*, 47–56. Springer, Berlin, Germany.

M. R. Garey and D. S. Johnson (1979). *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, NY.

J. Garnier and L. Kallel (1999). How to detect all maxima of a function? Technical report. to appear.

J. Garnier, L. Kallel, and M. Schoenauer (1999). Rigorous hitting times for binary mutations. *Evolutionary Computation 7*(2), 173–203.

D. E. Goldberg (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA.

O. Goldreich, S. Goldwasser, and D. Ron (1998). Property testing and its connections to learning and approximation. *Journal of the ACM 45*(4), 563–750.

R. B. Heckendorn and D. Whitley (1999). Predicting epistasis from mathematical models. *Evolutionary Computation 7*(1), 69–101.

D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis (1988). How easy is local search? *Journal of Computer and System Sciences 37*, 79–100.

T. Jones (1995). *Evolutionary Algorithms, Fitness Landscapes and Search*. Ph. D. thesis, University of New Mexico, Alburquerque, NM.

T. Jones and S. Forrest (1995). Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In L. J. Eshelman (Ed.), *Proceedings of the 6th International Conference on Genetic Algorithms (ICGA '95)*, 184–192. Morgan Kaufman, San Mateo, CA.

S. A. Kauffman (1993). *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, Oxford.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi (1983). Optimization by simulated annealing. *Science 220*, 671–680.

B. Naudts (1998). *Measuring GA-hardness*. Ph. D. thesis, University of Antwerp, Department of Mathematics and Computer Science, Antwerpen, Belgium.

B. Naudts and L. Kallel (1999). Comparison of summary statistics of fitness landscapes. *IEEE Transactions on Evolutionary Computation*. to appear.

B. Naudts, D. Suys, and A. Verschoren (1997). Epistasis as a basic concept in formal landscape analysis. In T. Bäck (Ed.), *Proceedings of the 7th International Conference on Genetic Algorithms (ICGA '97)*, 65–72. Morgan Kaufman, San Francisco, CA.

C. H. Papadimitriou, A. A. Schäffer, and M. Yannakakis (1990). On the complexity of local search (extended abstract). In *ACM Symposium on Theory of Computing*, 438–445.

R. J. Quick, V. J. Rayward-Smith, and G. D. Smith (1998). Fitness distance correlation and ridge functions. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel (Eds.), *Parallel Problem Solving From Nature (PPSN V)*, 77–86. Springer, Berlin, Germany.

G. Rudolph (1997). *Convergence Properties of Evolutionary Algorithms*. Verlag Dr. Kovač, Hamburg, Germany.

R. K. Thompson and A. H. Wright (1996). Additively decomposable fitness functions. Technical report, University of Montana, Computer Science Department.

E. D. Weinberger (1996). NP completeness of Kauffman's n-k model, a tuneably rugged fitness landscape. Technical Report SFI-TR-96-02-003, The Sante Fe Institute, Santa Fe, NM.

D. H. Wolpert and W. G. Macready (1997). No free lunch theorem for optimization. *IEEE Transactions on Evolutionary Computation 1*(1), 67–82.