A Note on Problem Difficulty Measures in Black-Box Optimization: Classification, Realizations and Predictability

Jun He j.he@cs.bham.ac.uk
School of Computer Science, University of Birmingham
Edgbaston, Birmingham B15 2TT, UK

Colin Reeves c.reeves@coventry.ac.uk
Department of Mathematical Sciences, Coventry University
Coventry CV1 5FB, UK

Carsten Witt cw01@ls2.cs.uni-dortmund.de
FB Informatik LS2, University of Dortmund, 44221 Dortmund, Germany

Xin Yao x.yao@cs.bham.ac.uk
School of Computer Science, University of Birmingham
Edgbaston, Birmingham B15 2TT, UK

Abstract

Various methods have been defined to measure the hardness of a fitness function for evolutionary algorithms and other black-box heuristics. Examples include fitness landscape analysis, epistasis, fitness-distance correlations etc., all of which are relatively easy to describe. However, they do not always correctly specify the hardness of the function. Some measures are easy to implement, others are more intuitive and hard to formalize.

This paper rigorously defines difficulty measures in black-box optimization and proposes a classification. Different types of realizations of such measures are studied, namely exact and approximate ones. For both types of realizations, it is proven that predictive versions that run in polynomial time in general do not exist unless certain complexity-theoretical assumptions are wrong.

Keywords

Black-box optimization, evolutionary algorithm, problem difficulty measure, running time, satisfiability problem.

1 Introduction

How can we characterize which fitness functions are easy for a given search heuristic and which are not? This problem was thought to be a major challenge especially in the field of evolutionary computation (Naudts and Kallel, 2000). It has attracted researchers' interests for over a decade, but as yet no satisfactory measure seems to have been found.

In this paper, we identify two approaches to state the hardness of a problem for a given search heuristic: the first one is to describe the characteristics of a fitness landscape, i. e., which characteristics of a fitness landscape make it hard for the heuristic and which do not. The second is to define difficulty measures using the values of the

fitness function. Often, these measures are estimated based on empirical observations using stochastic/statistical approaches, e. g., samples from the search space.

The first approach emerged in the early studies of fitness landscapes, where isolation, deception and multi-modality were linked to hard problems (Goldberg, 1989; Deb and Goldberg, 1993; Horn and Goldberg, 1995; Vose and Liepins, 1991; Forrest and Mitchell, 1993). Naudts and Kallel (2000) make the following observations: It is true that a fitness landscape with isolation (needle-in-a-haystack) is hard for evolutionary algorithms, but other characteristics may not be related too much to problem difficulty. For example, Vose and Wright (1995) present a fully non-deceptive function which is difficult for an evolutionary algorithm; Wilson (1991) proposes some deceptive functions that can easily be solved by evolutionary algorithms; Horn and Goldberg construct both an easy-to-solve multi-modal function (Horn and Goldberg, 1995), and a unimodal function—the 'long path' problem (Horn et al., 1994)—which is difficult for certain evolutionary algorithms, but easy for others (Rudolph, 1996).

The difficulty of fitness landscapes has also been described in terms of the concepts of ruggedness and neutrality (Smith et al., 2002). Recently, the difficulty of fitness landscapes has also been studied by means of fitness distributions and information landscapes (Borenstein and Poli, 2004, 2005). The concepts summarized thus far have in common that they provide intuitive explanations for the difficulty of a fitness landscape; however, they do not explicitly quantify difficulty as a measure, i. e., by numerical values. Thus they often evade a formal treatment.

As mentioned above, alternative approaches suggest that the difficulty of a problem should be computed by measures based on the fitness function and structure of the search space. Examples include fitness- distance correlation (Jones and Forrest, 1995), correlation length and operator correlation (Manderick et al., 1991), fitness variance (Radcliffe and Surry, 1995), and epistasis variance (Davidor, 1991). Jansen (2001) points out that the effort to compute *exactly* many of the measures, including epistasis variance and fitness distance correlation, in general is exponential in the problem size since the whole search space has to be explored. This immediately explains why, in practice, a statistical approach is taken and these measures are estimated using samples from the search space. Hence, as in the case of statistics, where theoretical and empirical versions of moments (e. g., mean values) exist, theoretical and empirical versions of difficulty measures should be carefully distinguished. This is also supported by Naudts and Kallel (2000) who separate so-called exact and approximate values of difficulty measures.

It is already known that even exact computations of many difficulty measures can be very misleading. Jansen (2001) defines example functions that coincide exactly in terms of their fitness distance correlation and epistasis variance whereas a simple evolutionary algorithm exhibits a completely different behavior on these functions. Jansen (2001) proves that the expected running time of this algorithm can vary from polynomial to exponential values even if the difficulty measure does not change at all.

Approximate computations of difficulty measures—in particular epistasis variance and fitness-distance correlation—are studied by Naudts and Kallel (2000). They demonstrate that using a sampling approach to compute average values of these measures can lead to very different outcomes compared to the exact values even if the problem is obviously easy for the considered search heuristic. This motivates the search for more robust measures that are reliable at least with a high probability. Naudts and Kallel (2000) also propose to define concepts of similarity for the behavior of search

heuristics. In an extremely simple case, this could mean that polynomial and superpolynomial running times of the search heuristics have to be distinguished.

Further Reeves (1999) points out inherent flaws in the common difficulty measures such as epistasis variance, fitness-distance correlation and epistasis correlation. He also raises an important theoretical question: can we know the difficulty of a problem without exploring the whole universe (i. e., the fitness of all points in the search space)? In other words, does a reliable and efficiently computable difficulty measure exist?

Based on the previous considerations and on Reeves's question, this paper describes a classification of problem difficulty measures along with possible realizations and then investigates their existence and predictability. The aim of the paper is to prove rigorously that to find a useful difficulty measure in general will be impossible. The rest of this paper is organized as follows. Section 2 provides the necessary definitions for the study of general search heuristics. Section 3 proposes a definition of difficulty measures and a classification thereof. Section 4 shows the main result that it is too optimistic to look for a general difficulty measure if we want to compute this measure efficiently. Finally, conclusions are drawn in Section 5.

2 Black-Box Algorithms and Running Times

In this paper, we concentrate on black-box algorithms, a model that comprises many randomized search heuristics including evolutionary algorithms. We follow the definition by Droste et al. (2006) but restrict ourselves to the cases of pseudo-boolean optimization problems, i. e., functions $f\colon \{0,1\}^n \to \mathbb{R}$. Without loss of generality, the aim is to maximize f.

Definition 1 (Black-Box Algorithm)

1. *Choose some probability distribution p on $\{0,1\}^n$ and produce a random search point $x_1 \in S$ according to p. Compute $f(x_1)$.*
2. *In step t, stop if the considered stopping criterion is fulfilled. Otherwise, depending on $I(t) = (x_1, f(x_1), \ldots, x_{t-1}, f(x_{t-1}))$, choose some probability distribution $p_{I(t)}$ on S and produce a random search point $x_t \in S$ according to $p_{I(t)}$. Compute $f(x_t)$.*

A crucial property of black-box algorithms is that they do not have any explicit access to the function for which an optimum is sought. The only way to obtain information on the unknown function is by evaluating different search points. This model is called the *black-box scenario* and is well accepted in the theory of optimization. Usually, an evaluation of the fitness function is the most costly part in optimization. This motivates the following definition.

Definition 2 (Running Time) *Let a black-box algorithm α and a fitness function $f\colon \{0,1\}^n \to \mathbb{R}$ be given. The* running time *$T_\alpha(f)$ of α on f is defined as the smallest t such that $f(x_t) = \max\{f(x) \mid x \in \{0,1\}^n\}$.*

The definition of the running time does not imply that the algorithm stops when an optimum has been evaluated. This is a consequence of the black-box scenario, which does not permit the algorithm to know that is has seen an optimum (except if the whole search space has been explored).

Obviously, the running time of a black-box algorithm in general is a random variable. Its expectation $E(T_\alpha(f))$, called the *expected running time* of α on f, is usually taken as a measure of difficulty of f for α. As usual in complexity theory (Papadimitriou and Steiglitz, 1998), running times that are polynomially bounded by the input size are taken as easy.

Definition 3 (α-hard/-easy) *A function $f\colon \{0,1\}^n \to \mathbb{R}$ is called* easy *w. r. t. α if $E(T_\alpha(f))$ is bounded by a polynomial in n and* hard *otherwise.*

A special case of a black-box algorithm is the famous (1+1) EA, which has already been considered extensively with respect to its expected running time (Droste et al., 2002; He and Yao, 2003). The (1+1) EA is especially easy to analyze since in each iteration, exactly one f-evaluation takes place.

```
begin
  generation counter t := 0;
  choose x_0 ∈ {0,1}^n uniformly at random;
  while (stopping criterion does not hold) do
        create y by flipping each bit of x_t uniformly at random;
        if f(y) ≥ f(x_t) then x_{t+1} := y;
                        else x_{t+1} := x_t;
        fi
        t := t + 1;
  end
end
```

The running time of the (1+1) EA is by 1 larger than the number of iterations until an optimal search point has been found. For general population-based evolutionary algorithms, the number of generations until an optimum has been found is typically less than the number of f-evaluations since several new individuals have to be considered in a generation. However, if the population size is bounded by a polynomial, running time and number of generations should differ only by a polynomial factor. The definition of hardness is invariant under such polynomial operations, and we can study the number of generations instead of f-evaluations for population-based EAs if we prefer.

In the following, we consider the decision variant of the combinatorial optimization problem called SAT: given a set of clauses $C_1, \cdots, C_m$ on the Boolean variables $x_1, \cdots, x_n$, determine whether the formula $C_1 \wedge \cdots \wedge C_m$ is satisfiable. The satisfiability problem is chosen here since the problem is a paradigmatic problem in the NP-completeness theory, and it is easy to polynomially transform the problem into other well-known NP-complete problems (see Chapter 15 in Papadimitriou and Steiglitz, 1998). Although EAs can be applied to solve SAT problems (e. g., Gottlieb et al., 2002; Michalewicz, 1996), here we do not try to carry out a theoretical running time analysis of the (1+1) EA and similar black-box heuristics for the NP-hard SAT problem. Note, however, that it is nowadays possible to analyze the (1+1) EA on NP-hard problems such as the partition problem (Witt, 2005).

The fitness function obtained from a SAT instance can be defined as the number of satisfied clauses; however, there may be more "EA-friendly" functions, see Gottlieb et al. (2002). In the following, we will instead use a very simple fitness function to show negative results.

3 A Classification of Difficulty Measures and Their Realizations

As mentioned in the introduction, the goal is to define measures that reliably indicate the difficulty of a function f for a black-box algorithm α, for example the (1+1) EA. If such a measure exists, its values allow us to distinguish hard and easy inputs with respect to the black-box algorithm. As an example, high epistasis is supposed to be linked to hard problems. Using the definition of hardness from Definition 3, we propose the following definition of a difficulty measure.

Definition 4 (Difficulty Measure) *Let a black-box algorithm α for the optimization of pseudo-boolean functions $f\colon \{0,1\}^n \to \mathbb{R}$ be given. A* difficulty measure for α *is a real-valued function M_α on the set of pseudo-boolean functions where there exists a threshold function $\theta(n)$ such that*

$$f \text{ is easy w.r.t. } \alpha \iff M_\alpha(f) \leq \theta(n). \tag{1}$$

The running time $\tau_\alpha(f)$ of α as a function of f and n is always a difficulty measure since we can define $\theta(n)$ as the supremum of all $p(n)$ such that $p(n)$ is a polynomial in n. However, this measure in general cannot be computed efficiently. We are interested in efficient, i. e., polynomial, realizations of difficulty measures. As mentioned in the introduction, it might be too ambitious to hope for efficient realizations that are always exact. This does not exclude that an approximate computation of measures based on samples from the search space is reliable at least with a high probability. We therefore also define randomized realizations that are allowed to fail (e. g., by giving wrong answers) with some hopefully small failure probability. In terms of complexity theory, we allow two-sided errors. This is linked to the notion of BPP (bounded-error, probabilistic, polynomial time) algorithms (Wegener, 2005). It is well known that the success probability of such an algorithm should be a constant greater than $1/2$, e. g., $3/4$, for the algorithm to be useful. Reusing the terminology of "exact vs. approximate" by Naudts and Kallel (2000), this leads us to the following definitions.

Definition 5 (Realizations of Difficulty Measures) *A deterministic algorithm that computes a given difficulty measure M_α is called an* exact realization. *A randomized algorithm that satisfies property* (1) *of Definition 4 with probability at least $3/4$ is called an* approximate realization. *Moreover, the realization is called* predictive *if its worst-case running time is bounded by a polynomial in n.*

It is by no means clear whether deterministic realizations always exist. If the black-box algorithm is any Turing machine program, we might be faced with an undecidable (i. e., non-recursive) problem since we have to decide uniformly for all problem sizes n whether the running time is bounded by a polynomial. Things turn out to be easier if we consider the simple (1+1) EA as optimizer. However, we will see that no predictive exact and approximate realizations exist unless P $=$ NP or BPP $=$ NP. (The class BPP consists of the problems for which a BPP algorithm exists.) This means that all functions that are meant to be difficulty measures according to Definition 4 (including possible improvements of epistasis variance etc.), do not allow predictive realizations.

Summarizing the above discussion, we get a classification of realizations of difficulty measures in Figure 1. Note that for technical reasons, each exact realization is also approximate according to Definition 5.

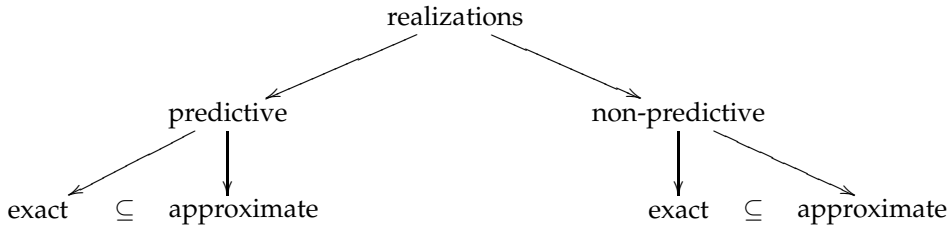

Figure 1: A classification of realizations of difficulty measures.

4 Predictive Measures Probably Cannot Be Realized

In this section, we show that exact (resp. approximate) predictive measures in general cannot be realized unless P = NP (resp. BPP = NP). This is achieved by a reduction from the SAT problem introduced above.

We have the freedom to choose the black-box algorithm α for which we want to consider a predictive difficulty measure. For simplicity, we take the (1+1) EA. The idea is to transform satisfiable instances of the SAT problem into so-called almost-needle functions. Such functions have value 0 on at least a $(1 - 2^{-\Omega(n)})$-fraction of its inputs and value 1 on at least one input. It is easy to see that the expected running time of the (1+1) EA on almost-needle functions is exponential (Wegener and Witt, 2005). On the other hand, unsatisfiable instances will be transformed into functions that take value 0 on all inputs. We have to ascertain that such a transformation from the SAT problem is possible in polynomial time. This means that we must be able to describe in polynomial time an algorithm that computes the fitness function.

Let a SAT formula f with m clauses on n variables $x_1, \ldots, x_n$ be given. We start from the fitness function $s\colon \{0,1\}^n \to \mathbb{R}$ such that $s(x) = 1$ if $x = (x_1, \ldots, x_n)$ is a satisfying assignment for f and $s(x) = 0$ otherwise. Obviously, an algorithm for s can be described in polynomial time with respect to n and m. (A run of this algorithm, i. e., an evaluation of s, can also be made possible in polynomial time with respect to n, but this does not matter in the black-box scenario.) The fitness function s is not necessarily a needle-like function, it might happen that all assignments are satisfying. Therefore, we transform s into an almost-needle function s' using a padding technique as follows: We introduce n additional variables $y_1, \ldots, y_n$ and define the pseudo-boolean function $s'\colon \{0,1\}^{2n} \to \mathbb{R}$ on the original and additional variables as follows:

$$s'(x_1, \ldots, x_n, y_1, \ldots, y_n) := \begin{cases} 1 & \text{if } s(x_1, \ldots, x_n) = 1 \text{ and } y_1 = \cdots = y_n = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Obviously, also an algorithm for s' can be described (and run, which again does not matter) in polynomial time with respect to n and m. The function s' is an almost-needle function since it is 1 for at most 2^n of 2^{2n} possible inputs. By definition, it differs from the constant function 0 if and only if the formula f has a satisfying assignment. Hence, the expected running time of the (1+1) EA on s' is polynomial (in fact it is constant) if and only if f is unsatisfiable.

Suppose now there is a predictive exact realization of a difficulty measure for the (1+1) EA. In particular, we can apply this measure to s' and check in polynomial time whether the (1+1) EA will have polynomial or non-polynomial running time on s'. This corresponds to deciding in polynomial time whether f is satisfiable. For general SAT formulas, this is impossible unless P = NP (Wegener, 2005).

With respect to predictive approximate realizations of difficulty measures, the same construction applies. If such a realization exists, it will with probability at least $3/4$ decide in polynomial time whether f is satisfiable. This is impossible unless BPP = NP (Wegener, 2005).

We conclude that, from a worst-case perspective in terms of all possible fitness functions, we cannot hope to build predictive difficulty measures. These can however exist if the considered EA is applied to problems that do not have any hard instances. This will be discussed briefly in the following section.

5 Conclusions

This paper gives a rigorous definition of difficulty measures in black-box optimization. Motivated by the distinction between theoretical and empirical versions of statistical measures, we have classified realizations of difficulty measure into two types: namely exact and approximate realizations. Assuming a worst-case perspective, the paper has rigorously proven for both types that predictive versions, i. e., polynomial-time implementations, do not exist unless P = NP or BPP = NP.

An interesting question for future research is: Given an algorithm for the SAT problem or other NP-hard problems, can we design a predictive measure for a broad class of the corresponding fitness functions? Of course, if the class of fitness functions includes only several special easy instances, e. g., in the SAT problem, any formula without negated variables, the answer is yes. However, if the class is large and is a non-trivial subset of the whole set of instances, the answer is still unclear.

Acknowledgment

An early version of the paper was presented on the PPSN IX Workshop on Evolutionary Algorithms - Bridging Theory and Practice. We would like to thank the anonymous reviewers for their valuable suggestions. This work is supported by the UK Engineering and Physical Sciences Research Council under Grant (EP/C520696/1). Financial support by the Deutsche Forschungsgemeinschaft (DFG) in terms of the Collaborative Research Center "Computational Intelligence" (SFB 531) is gratefully acknowledged.

References

Borenstein, Y. and Poli, R. (2004). Fitness distributions and GA hardness. In *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN 2004)*, pages 11–20. Springer.

Borenstein, Y. and Poli, R. (2005). Information landscapes and problem hardness. In *Proceedings of the 2005 Genetic and Evolutionary Computation Conference (GECCO 2005)*, pages 1425–1431. ACM Press.

Davidor, Y. (1991). Epistasis variance: A viewpoint on GA-hardness. In Rawlins, G. J. E., editor, *Foundations of Genetic Algorithms (FOGA)*, pages 23–35, San Mateo, CA, Morgan Kaufmann.

Deb, K. and Goldberg, D. E. (1993). Sufficient conditions for deceptive and easy binary functions. *Annals of Mathematics and Artificial Intelligence* 10(4):385–408.

Droste, S., Jansen, T., and Wegener, I. (2002). On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science 276*, 51–81.

Droste, S., Jansen. T., and Wegener, I. (2006). Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of Computing Systems* 39(4):525–544.

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

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

Gottlieb, J., Marchiori, E., and Rossi, C. (2002). Evolutionary algorithms for the satisfiability problem. *Evolutionary Computation* 10(1):35–50.

He, J. and Yao, X. (2003). Towards an analytic framework for analysing the computation time of evolutionary algorithms. *Artificial Intelligence* 145(1-2):59–97.

Horn, J. and Goldberg, D. E. (1995). Genetic algorithm difficulty and the modality of fitness landscapes. In Whitley, L. D. and Vose, M. D., editors, *Foundations of Genetic Algorithms (FOGA) 3*, pages 243–269, San Mateo, CA, Morgan Kaufmann.

Horn, J., Goldberg, D. E., and Deb, K. (1994). Long path problems. In *Proceedings of the 3rd International Conference on Parallel Problem Solving from Nature (PPSN 1994)*, pages 149–158, Springer.

Jansen, T. (2001). On classifications of fitness functions. In L. Kallel, B. Naudts and A. Rogers (Eds:), *Theoretical Aspects of Evolutionary Computing*, pages 371–386. Springer.

Jones, T. and Forrest, S. (1995). Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In Eshelman, L. J., editor, *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 184–192, San Mateo, CA, Morgan Kaufmann.

Manderick, B., de Weger, M., and Spiessens, P. (1991). The genetic algorithm and the structure of the fitness landscape. In Belew, R. K. and Booker, L. B., editors, *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 143–150, San Mateo, CA. Morgan Kaufmann.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structure = Evolution Programs*. Springer Verlag, New York, third edition.

Naudts, B. and Kallel, L. (2000). A comparison of predictive measure of problem difficulty in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* 4(1):1–15.

Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial Optimization: Algorithms and Complexity*. Dover, Mineola, NY.

Radcliffe, N. J. and Surry, P. D. (1995). Formae and the variance of fitness. In Whitley, L. D. and Vose, M. D., editors, *Foundations of Genetic Algorithms (FOGA) 3*, pages 51–72, San Mateo, CA., Morgan Kaufmann.

Reeves, C. R. (1999). Predictive measures for problem difficulty. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC 1999)*, pages 736-743, IEEE Press.

Rudolph, G. (1996). How mutation and selection solve long path problems in polynomial expected time. *Evolutionary Computation* 4(2):194–205.

Smith, T., Husbands, P., Layzell, P. J., and O'Shea, M. (2002). Fitness landscapes and evolvability. *Evolutionary Computation* 10(1):1–34.

Vose, M. D. and Liepins, G. E. (1991). Punctuated equilibria in genetic search. *Complex Systems*,5:31–44.

Vose, M. D. and Wright, A. H. (1995). Stability of vertex fixed points and applications. In Whitley, L. D. and Vose, M. D., editors, *Foundations of Genetic Algorithms (FOGA) 3*, pages 103–114, San Mateo, CA, Morgan Kaufmann.

Wegener, I. (2005). *Complexity Theory – Exploring the Limits of Efficient Algorithms*, Springer.

Wegener, I. and Witt, C. (2005) On the optimization of monotone polynomials by simple randomized search heuristics. *Combinatorics, Probability & Computing* 14:225-247.

Wilson, S. W. (1991). GA-easy does not imply steepest-ascent optimizable. In Belew, R. and Booker, L., editors, *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 85–89, San Mateo, CA. Morgan Kaufmann.

Witt, C. (2005). Worst-case and average-case approximations by simple randomized search heuristics. In: *Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS 2005)*, pages 44–56, LNCS 3404, Springer.