Fitness Distributions and GA Hardness

Yossi Borenstein and Riccardo Poli

Department of Computer Science, University of Essex
{yboren,rpoli}@essex.ac.uk

Abstract. Considerable research effort has been spent in trying to formulate a good definition of GA-Hardness. Given an instance of a problem, the objective is to estimate the performance of a GA. Despite partial successes current definitions are still unsatisfactory. In this paper we make some steps towards a new, more powerful way of assessing problem difficulty based on the properties of a problem's fitness distribution. We present experimental results that strongly support this idea.

1 Introduction

For over a decade GA researchers have attempted to predict the behavior of a GA in different domains. The goal is to be able to classify problems as hard or easy according to the performance a GA would be expected to have on such problems, accurately and *without actually running the GA*.

The Building Block (BB) hypothesis [1] states that a GA tries to combine low, highly fit schemata. Following the BB hypothesis the notion of deception [1], [2] isolation [3] and multimodality [4] have been defined. These were able to explain a variety of phenomena. Unfortunately, they didn't succeed in giving a reliable measure of GA-hardness [5], [6].

Given the connection between GAs and theoretical genetics, some attempts to explain the behavior of GAs were inspired by biology. For example, epistasis variance [7] and epistasis correlation [8] have been defined in order to estimate the hardness of a given real world problem. NK landscapes [9], [10] use the same idea (epistasis) in order to create an artificial, arbitrary, landscape with a tunable degree of difficulty. These attempts, too, didn't succeed in giving a full explanation of the behavior of a GA [6], [11], [12].

Finally, fitness distance correlation [13] tries to measure the intrinsic hardness of a landscape, independently of the search algorithm. Despite good success, fitness distance correlation is not able to predict performance in some scenarios [14].

The partial success of these approaches is not surprising. Several difficulties present themselves when developing a general theory that explains the behavior of a GA and is able to predict how it will perform on different problems. These include:

- A GA is actually a *family* of different algorithms. Given a problem the GA designer first decides which representation (e.g. binary, multiary, permutation, real numbers) to use, then how to map the solution space into the search space, and finally which operator(s) (mutation, crossover) to use. Moreover, there are limited concrete guidelines on how to choose a representation and a genotype-phenotype

X. Yao et al. (Eds.): PPSN VIII, LNCS 3242, pp. 11–20, 2004.

mapping. Indeed this is a very difficult task. Different genotype-phenotype representations can completely change the difficulty of a problem [15]. There have been attempts to evolve the right representation [16] and there are some general design guidelines [15], [17], [18]. However, the reality is that the responsibility of coming up with good ingredients for a GA is still entirely on the GA designer.

- According to [19] it is impossible to predict the performance of an algorithm based only on the on the *description* of the problem instance and the algorithm without actually running the algorithm.

In the absence of a good, predictive theory of GA performance, unavoidably we are only left with an experimental approach. The idea is to divide the space of real-world problems into GA hard and GA easy by finding (by experimentation and experience) a good GA (with its representation, mapping, and operators) for every specific instance (or class of instances) of a problem.

The No Free Lunch (NFL) theorem [20] states that, on average, over all possible problems, the performance of each search algorithms is equal. The basis for this claim is the observation that any prediction based on sampling a sub-space may not be valid. By studying some limitations that general knowledge of a problem might put on this observation, we introduce a new way to classify problems for GAs. In particular we explore assessing problem difficulty based on the properties of a problem's fitness distribution.

In the next sections we introduce our notation, we explain our motivation for the new classification idea, define it and give some empirical results that support it. We conclude with a discussion about possible implications and future work.

2 Definition

2.1 Notation

Following [21] we use the following notation. A **problem** is a fitness function $f : X \to Y$ that we would like to maximize (where X is the search space and Y is the ordered set of fitness values). This assigns a fitness value to each point in the search space. A **representation** is the way that we choose to represent the search space (i.e. real numbers, binary strings). We define a point in the search space as a **phenotype,** the chosen representation for it as a **genotype,** and the function that transforms the genotype into the phenotype as a **genotype-phenotype mapping**.

If we sampled the search space randomly and we recorded the fitness of the solutions sampled, different fitness values (i.e. different elements of Y) would have potentially different probabilities of occurrence. We define the **fitness distribution** $p(y)$ as the frequency of each particular $y \in Y$ for the problem of interest f. For example, the fitness distribution for a one-max problem is (assuming $X=\{0,1\}^N$):

$$p(y) = \binom{N}{y} . 2^{-N} \tag{1}$$