# Information and Performance Landscapes

Yossi Borenstein and Riccardo Poli
Department of Computer Science
University of Essex, UK
{yboren,rpoli}@essex.ac.uk

Department of Computer Science
University of Essex
Technical Report CSM-440
ISSN: 1744-8050
November 2005

**Abstract**

When no knowledge is available on the structure of the fitness landscape, effectively the fitness function is a black box. Search in this scenario is not well understood. In this paper we introduce a new theoretical approach to study search algorithms in this scenario. This is based on a new definition of the concept of landscape – the information landscape – which allows us to evaluate the *quantity* of information embedded in a landscape. We derive a linear approximation to the performance of an algorithm based on this notion. This leads to the definition of the concept of performance landscape, which allows us to evaluate the *quality* of the information in a landscape with respect to an algorithm's search behaviour. We show that these two landscapes and their simple interaction allow us to better understand the general behaviour of search algorithms in the black box scenario. We also derive a new simple measure of problem difficulty. In experiments with a large variety of test problems and different versions of genetic algorithm we provide strong evidence supporting this framework.

## 1 Introduction

In the last 30 years many algorithms (also known as metaheuristics) have been proposed to explore search spaces in an efficient way (Blum and Roli, 2003). Many of these are inspired by powerful natural or physical processes. For example, Ant Colony Optimisation (Dorigo and Stützle, 2004) is inspired by the food foraging behaviour of ants, Simulated Annealing (Kirkpatrick et al., 1983) is inspired by the annealing process of metals and glass, and, of course, Evolutionary Algorithms (EAs) (Fogel et al., 1966; Holland, 1975; Rechenberg, 1973; Schwefel, 1981) are inspired by the processes of natural selection and genetics.

These and other metaheuristics have been applied successfully to an ever increasing number of hard functional and combinatorial optimisation problems. However, in many cases, their remarkable empirical success is not associated with corresponding success in developing theoretical foundations with real practical ramifications. As a result it is still difficult to match efficiently algorithms with problems or even to tune the parameters of a search algorithm to better solve a specific class of problems.

Roughly speaking existing theoretical work usually adopts one of the following three approaches. The first approach studies the properties of particular algorithms. The objective is either to describe their dynamics or to compute concrete bounds to their expected performance (Nix and Vose, 1992; Prügel-Bennett and Shapiro, 1994; Stephens and Waelbroeck, 1999; He and Yao, 2003; Droste, 2004; Jansen and Wegener, 2000). Rather than studying each algorithm separately, the second approach focuses on their general aspects. The no-free-lunch theory (Wolpert and

Macready, 1997), for example, studies the limitation on search performance for an algorithm sampling a search space in the attempt to optimise a fitness function, on which, however, no knowledge is available (*black box scenario*). This is done without reference to any particular algorithm, and, so, any conclusions apply to all algorithms. Finally, the third approach (Jones, 1995a; Reidys and Stadler, 2002) assumes that the performance of different algorithms depends mainly on the connection between the neighbourhood structure and the fitness function. The combination of the two is referred to as *fitness landscape*. The objective there is to study properties of a landscape which are likely to affect the performance of search algorithms.

In the next subsections we give a short overview of the three approaches focusing, for the sake of brevity, on the Genetic Algorithm (GA). Being one of the most popular metaheuristics, the GA is an interesting case-study.

## 1.1 Algorithm-specific approaches

The dynamical system approach (Nix and Vose, 1992; Vose, 1998) and schema theory (Stephens and Waelbroeck, 1999; Poli et al., 2002) are successful examples of the explicit study of the exact properties of simple GAs or genetic programming systems (Langdon and Poli, 2002). Unfortunately, the application of such models to new search operators or different representations requires a significant effort and time. The constant emergence of new variants of EAs (e.g., variable length GAs (Lee and Antonsson, 2000), new biologically inspired algorithms (Simões and Costa, 2002) and redundant representations (Rothlauf, 2002)) makes it hard for this kind of theory to keep up with the state of the art.

Given the difficulty of analysing the combined effect of all the operators, many researchers have studied the effect of different operators separately. For example, (H"ohn and Reeves, 1996) studied the properties of crossover, finite population effects (drift) were studied in (Asoh and Mühlenbein, 1994), while more recently (Droste, 2004) and (Jansen and Wegener, 2000) presented some useful results on the time complexity and the optimal parameter settings for a mutation-based GA. Usually these approaches are successful, but often, in order to make progress, researchers are forced to limit their attention to simple artificial problems, like, for example, the famous counting-ones problem (also known as Onemax).

Another approach in this category is the study of the properties of problems that, given some knowledge on the algorithm, are expected to affect performance. For instance, the building block hypothesis (Goldberg, 1989) states that a GA solves problems by combining low, highly fit schemata. The related notion of deception (Goldberg, 1989; Forrest and Mitchell, 1992) was suggested as a key element for understanding what makes a problem hard for a GA. Epistasis variance (Davidor, 1990) and epistasis correlation (Naudts, 1998) incorporated knowledge from theoretical genetics in order to infer GA hardness. Sitewise optimisation (Naudts and Kallel, 2000) – a generalisation of fitness distance correlation – and epistasis have also been suggested for this task. These methods work well on many problems, but, unfortunately, none can be considered a fully reliable measure of GA-hardness (Naudts and Kallel, 2000; Grefenstette, 1992; Jansen, 2001; Guo and Hsu, 2003).

Fundamentally, the problems encountered by all these approaches are related to the intrinsic complexity of modern metaheuristics, which makes it difficult to explore their dynamics theoretically. Indeed, recent work published in the EC field suggests that, due to lack of proper general classification methods, empirical results should be used instead of theory (Jansen, 2001; Guo and Hsu, 2003).

Even if it was possible to explore each algorithm theoretically, these methods could give only limited insights about search in the black box scenario in general. The black-box scenario imposes many constraints on the performance of algorithms. Understating these constraints is an important, if not fundamental step to the understanding of any algorithm. This is the objective of the methods described in the next section.

## 1.2 Algorithm-independent approaches

Wolpert and Macready considered the general properties of the black box scenario (Wolpert and Macready, 1997). The results of their investigation, the No-Free-Lunch Theorems (NFLTs), put an end to the hope of developing a general-purpose robust optimisation algorithm. The NFLTs prove that such an algorithm does not exist.

In particular, the NFLTs consider the set of all possible cost functions of the form $f : X \to Y$, where $X$ – a search space – and $Y$ – the set of possible fitness values – are finite.[1] A generic deterministic optimisation algorithm $a : (X \times Y)^m \to X$ is represented as a mapping from a set of $m$ previously visited points and their associated fitness values to a single new (i.e., previously unvisited) point in $X$.

Randomised search heuristics, unlike specialised algorithms, do not use problem-specific knowledge. They sample possible solutions, compute their objective values and accordingly sample additional solutions. This continues until some stopping criterion is met. The following general algorithm (see (Droste et al., 2003; Wegener, 2003; Wegener, 2004)) generalises most, if not all, existing randomised search heuristics:

1. Choose some probability distribution $p$ on $X$ and produce a random search point $x_1 \in X$ according to $p$. Compute $f(x_1)$.

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

We term this algorithm a *black box algorithm*.[2]

Wolpert and Macready proved that averaged over all possible algorithms, all problems have the same difficulty, and, averaged over all possible problems, all algorithms have the same efficiency (Wolpert and Macready, 1997).

Following this line of research, English (English, 2004; English, 2000a; English, 2000b) connected the notion of Kolmogorov complexity to the black box scenario, while Schumacher, Vose and Whitley (Schumacher et al., 2001) proved that the NFLTs hold for small subsets of problems as well (as long as the set is closed under permutation).

Despite this excellent progress, the connection between the set of problems for which the NFLTs hold and real-world problems remains very unclear. (Culberson, 1998), for example, argued that real-world problems contain information (structure) which is not available in the black box scenario and, hence, the practical implications of these results for realistic problems and search algorithms are not clear.

## 1.3 Fitness landscape approaches

The notion of fitness landscape was introduced in theoretical genetics (Wright, 1932; Weinberger, 1991; Reidys and Stadler, 2002) as a way to visualise evolution dynamics. The intuition behind it is that fitness can be thought of as a height function, orthogonal to the genome space and hence high fitness points are located in peaks and low fitness points in valleys. This notion has been later introduced in evolutionary computation (Jones, 1995a).

The fitness landscape approach can be thought of as being half way between the algorithmic specific approach and the NFLTs one. Similarly to the NFLTs, it does not consider any specific search algorithm. However, it assumes that all the search operators used by the different algorithms are defined over the neighbourhood structure. As a consequence, performance is mainly dependent on the connection between the neighbourhood structure and the fitness function.

Isolation (Goldberg, 1993) and multimodality (Rana, 1998) are perhaps the most intuitive features of a landscape that may be associated with hardness. Isolation might be a sufficient

[1] All objective functions implementable in a digital computer are of this form.

[2] This is the unrestricted version given in (Droste et al., 2003).

condition for a difficult landscape but it is certainly not necessary. Multimodality is neither necessary nor sufficient for a landscape to be difficult (Grefenstette, 1992; Jansen, 2001).

Fitness distance correlation (Jones and Forrest, 1995) measures the hardness of a landscape according to the correlation between the distance from the optimum and the fitness of solutions. Despite being generally successful, fitness distance correlation is not able to predict performance in all scenarios (Altenberg, 1997a; Jansen, 2001).

NK landscapes (Altenberg, 1997b; Kauffman, 1993) use the idea of epistasis in order to create arbitrary artificial landscapes with a tunable degree of difficulty. Even though NK landscapes are interesting from theoretical genetics point of view, their practical use has been questioned (Jansen, 2001).

## 1.4 Critique of the fitness landscape approach

To the best of our knowledge, at present, there is no measure of landscape difficulty that is able to capture precisely the hardness of the search. In our opinion, there are various reasons why linking the properties of a fitness landscape with search hardness is problematic.

Firstly, it is arguable that the main focus when considering a fitness landscape should be the information that is available to the search algorithm in order to conduct the search. The needle-in-a-haystack (NIAH) problem, for example, is considered to be very difficult because it does not contain any information to guide the search. Intuitively, the *quantity of information* available to an algorithm is very important in determining its performance. Naturally, this is not the only factor – even if there is abundant information in a landscape, this information can be such to deceive an algorithm, steering it away from global optima more often than not. The *quality of the information* available to an algorithm is naturally crucial in determining performance. A limitation of the original fitness landscape approach is that it does not distinguish between the two. In this paper, thanks to a *redefinition of the notion of landscape*, we provide a *formal* way to quantify the amount of information available in a landscape and to asses its quality.

Secondly, the analysis of the fitness landscape assumes that the neighbourhood structure and the fitness function are the only properties relevant to the performance of a search algorithm – that is, it assumes not only that search operators are defined over the neighbourhood structure but also that their exact formulation is not important. (Jones, 1995b; Reeves, 1999) argue, on the other hand, that this is not the case – different search algorithms use different operators which induce *different landscapes*. From this perspective, evaluating the difficulty of the search without considering the details of the algorithm being used seems hard if not impossible. Following a similar reasoning, Stadler and Stephens state:

> "Hill climbing on a static fitness landscape is an adequate metaphor in the case of evolution dominated by selection. We have emphasised in this paper that when this is not the case, then the utility of the landscape concept is much diminished." (P.F.Stadler and C.R.Stephens, 2002)

Finally, there may be redundant information in a fitness landscape, at least for some modern metaheuristics. For example, in GAs using tournament selection, the exact fitness of solutions is of no relevance to the algorithm: only the relative order between individuals is. So, analyses based on the full fitness landscape may end up being overly complicated.

## 1.5 This paper

To summarise the previous discussion: the NFLTs explore the general properties of algorithms in the black box scenario, but it is not clear how they can be applied to specific search algorithms; algorithm-specific theoretical models explain, in some scenarios, the behaviour of a specific search algorithm on a specific problem, but these results cannot easily be applied to other search algorithms (or other problems); finally, as we just discussed, using the concept of fitness landscape to assess problem difficulty is problematic.

In this paper we suggest a new theoretical approach to study search algorithms which overcomes some of the limitations of existing techniques. Similarly to NFLTs, our approach can explore properties of search in the black box scenario in general. However, it is not limited to that: it can also be applied to infer properties of specific search algorithms. Moreover, similarly to the fitness landscape approach, it can be used to assess problem hardness.

The paper is organised as follows. In the next section we give a general introduction to our approach, discussing its assumptions and possible limitations. In Section 3 we give a new definition of the concept of a landscape. Since this is based on the quantity of the information available to the search algorithm we call this landscape an *information landscape*. In section 4, our linear approximation to the performance function is computed. This leads to the definition of the concept of *performance landscape*. We present in Section 5 the implications of the new framework for search in the black box scenario. In Section 6 we argue that this framework can be used to provide a predictive measure to problem difficulty. Section 7 presents empirical results which support our theoretical arguments and shows how the framework can be used to infer properties of particular search algorithms. We conclude with a brief discussion and some final observations in Section 8.

# 2 Our approach

Consider a particular black-box search algorithm. The performance of the algorithm on a problem is a function of the form $P : X \times \mathcal{N} \times f \rightarrow \mathbb{R}$, where $X$ is the search space, $\mathcal{N}$ is the neighbourhood structure (which is defined over $X$) and $f : X \rightarrow \mathbb{R}$ is a fitness function. $P$ can be any given measure of search performance. Typically it is the number of queries (i.e., fitness evaluations) made until a sufficiently good solution is found although we do not restrict ourselves to this definition of performance (see (Wolpert and Macready, 1997) and (Droste et al., 2003) for further discussion).

As we will show in Section 4, our framework is based on a first order approximation to $P$. This is, surprisingly, sufficient to predict the performance of a GA over many problems. Moreover, it gives important insights for understanding search in the black box scenario. However, although this approach is very productive, we need to emphasise that most of the theoretical results are valid to a first order approximation.

In addition, our framework is based on the comparison of pairs of solutions in the search space (as we will discuss in Section 3). The method, at this stage, is therefore limited to search algorithms that use only the relative fitness (order) of solutions when deciding what to do next. This limitation is not severe, since many popular search algorithms indeed use only the relative order. For example, GAs with tournament selection or rank selection, binary particle swarm optimisers, hill climbers, simulated annealing, etc. can all be studied with our approach.[3]

If there is more than one global optimum or the objective of the search is not to find the global optimum but to find solutions of sufficiently good quality, the subset $X_{\text{target}}$ of solutions where the search algorithm will stop may contain more than one point. Given a search algorithm and a particular problem, an important objective of theoretical analysis would be to assess the probability that the search algorithm will find a member of $X_{\text{target}}$ in a particular number of fitness evaluations. Ideally such an analysis would enable us to associate each algorithm with a particular subset of problems on which it is expected to perform well. This, however, is an extremely hard task. So, instead of attacking this problem directly, in this paper we consider a simpler version of the problem. We will assume that $X_{\text{target}}$ contains only a single solution, $x_{\text{target}}$.[4] This has an important implication: for any given choice of $x_{\text{target}}$, our approach allows us to study the performance of algorithms on the set of all possible problems which have *the same global optimum* (i.e. $x_{\text{target}}$).

We need to distinguish between two types of algorithms. When an algorithm is unbiased

[3]Naturally, it is possible that other algorithms, such as GAs with fitness proportionate selection, may approximately be studied with this approach. However, whether or not the degree of approximation would be acceptable is something we need to study in future research.

[4]When this is not the case, our approach will give a lower bound to the actual efficiency of an algorithm (this is because the algorithm will not stop until it visits one specific member of $X_{\text{target}}$ rather than any member).

w.r.t. any region of the landscape, its performance is invariant to the position of $x_{\text{target}}$. That is, studying the performance of this type of algorithms on the subset of all problems with the same $x_{\text{target}}$ is sufficient in order to get a complete picture of the behaviour of the algorithm on all possible problems (we will formalise this in Section 5.2). Our approach is particularly well suited for this situation. A second class of algorithms includes algorithms that search preferentially certain regions of the search space. These are biased and so we do not have symmetry w.r.t. $x_{\text{target}}$. For these algorithms one could still apply our framework, but in order to get a complete picture one would have to apply it to all possible assignments of $x_{\text{target}}$ in $X$. (See Section 5.2 for a more formal analysis.)

Although is many cases one would want to stop the search algorithm when $x_{\text{target}}$ is sampled and use as a performance measure the number of fitness evaluations or generations required for the algorithm to do so, these are not requirements in our framework. For example, the framework is also applicable to algorithms that stop when the population has converged (ideally, to $x_{\text{target}}$) or to performance measures which depend on the state of the population as a whole (like, for example, the takeover time – see Section 7.1).

Having discussed the assumptions, features and limitations of our framework we are ready to introduce it more formally.

# 3 Information landscapes

An *information landscape* is a triple $(X, \mathcal{N}, t)$ including

1. a set $X$ of configurations,
2. a notion $\mathcal{N}$ of neighbourhood, nearness or distance on $X$, and
3. a stochastic information function $t : X \times X \to [0, 1]$ .

For every pair $(x_i, x_j)$ of elements in $X$, $t$ gives the probability that $x_i$ is superior to $x_j$ (i.e., the fitness of $x_i$ is higher than that of $x_j$). The value of the function $t$ can be viewed as the outcome of a stochastic tournament selection with tournament size two. Naturally, the function $t$ can be represented as an $|X| \times |X|$ information matrix $M$ with entries $m_{i,j} = t(x_i, x_j)$. Note that when $X$ is implied we can use the term information landscape to denote $M$ without ambiguity.[5] Figure 1 gives an example of a fitness function, a fitness landscape and the information matrix which represents our information landscape for a bit-string configuration space $X$.

When a fitness function $f$ is available, normally:

$$t(x_i, x_j) = \begin{cases} 1 & \text{if } f(x_i) > f(x_j), \\ 0.5 & \text{if } f(x_i) = f(x_j), \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

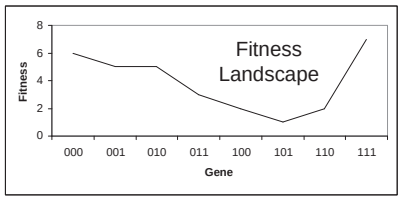
However, if the fitness function is noisy, $t$ can take values other than 0, 0.5 and 1. Thus, every fitness function has a corresponding information landscape. Conversely, given an information landscape we can construct the following rank-based fitness function:

$$f_{\text{rank}}(i) = \sum_j m_{i,j}. \tag{2}$$

It is important to note, however, that not all information landscapes can be associated to a fitness function (the information matrix does not necessarily represent a partial order). We will call *invalid* those information landscapes that cannot be derived from corresponding fitness landscapes.

As mentioned in Section 2, in this paper we consider only optimisation problems with one global optimum ($x_{\text{target}}$). Typically the objective of the search is to find it, although we do not

[5] In the rest of the paper we use the terms problem and landscape interchangeably, and, whenever we mention the term "landscape", we will mean an information landscape unless otherwise stated.




(b)

(a) Fitness Function

| Gene | Fitness |
|---|---|
| 000 | 6 |
| 001 | 5 |
| 010 | 5 |
| 011 | 3 |
| 100 | 2 |
| 101 | 1 |
| 110 | 2 |
| 111 | 7 |

(c) Information Landscape

| Gene | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| 000 | | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 001 | 0 | | 0.5 | 1 | 1 | 1 | 1 | 0 |
| 010 | 0 | 0.5 | | 1 | 1 | 1 | 1 | 0 |
| 011 | 0 | 0 | 0 | | 1 | 1 | 1 | 0 |
| 100 | 0 | 0 | 0 | 0 | | 1 | 0.5 | 0 |
| 101 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 |
| 110 | 0 | 0 | 0 | 0 | 0.5 | 1 | | 0 |
| 111 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |

Figure 1: Three ways of representing the information given to a search algorithm: a) a fitness function (represented here as a vector), b) a plot representing the fitness landscape, and c) a matrix representing the outcome of all possible comparisons between solutions (information landscape). The matrix presents symmetries with respect to the diagonal (see text) which reduce the number of available degrees of freedom; the gray area marks the independent elements of the information landscape.

necessarily have to stop the algorithm when that happens. We assume that the algorithm has a way of identifying the optimum (e.g., from its fitness) when this gets sampled. Note that the entries of the information landscape in the row and column corresponding to the global optimum are always the same irrespective of the problem (the unique global optimum is, by definition, always better than any other solution in the search space). For this reason we can exclude those entries from the information landscape without any loss of information (e.g., the rightmost column and the last row in Figure 1 (c) are excluded from the information landscape). Effectively, we can assume that the search space is the set $X' = X \setminus \{x_{\text{target}}\}$.

In addition, since $t(x_i, x_j) = 1 - t(x_j, x_i)$, the matrix $M$ presents symmetries with respect to the main diagonal which reduce the number of available degrees of freedom to the elements above (or, alternatively, below) the diagonal. The gray area in Figure 1 shows the independent elements of the information landscape. Diagonal elements (omitted for clarity) are all 0.5 since they are the result of comparing pairs of identical solutions. In order to represent an information landscape more concisely, we use the following vector to store the relevant (above diagonal) entries in the information matrix:

$$V = (v_1, v_2, ..., v_n) = (m_{1,2}, m_{1,3}, ..., m_{|X|-1,|X|}), \tag{3}$$

where $n = (|X| - 1) \times (|X| - 2)/2$.

## 3.1 Quantity of information

The information landscape framework provides an intuitive way to quantify the amount of information available in the landscape. As we mentioned in Section 1.2, most modern metaheuristics are essentially black-box algorithms. As a result they use the fitness function alone as an indicator to areas of the search space which are expected to contain fit solutions. Naturally, this can only be done if a reasonable proportion of the solutions in the search space have different fitness values. Otherwise, an algorithm will often be forced to select arbitrarily which solution to sample next.

Some algorithms (e.g., GAs with tournament selection) use only the relative fitness of solutions for the purpose of making a decision (e.g., choosing the winner of a tournament). In this case, whenever two solutions with identical fitness are compared, the algorithm will be forced to make a *random decision*. So, virtually irrespective of the algorithm, the more pairs of solutions with identical fitness there are, the more randomised the search will be.

In the information landscape, solutions with identical fitness are represented by matrix elements with a value of 0.5. Therefore, the proportion of elements of the information landscape for which a search algorithm could make an *informed* (i.e., non-random) decision is given by

$$d^{0.5}(V) = \frac{1}{n} \sum_i \delta(v_i \neq 0.5) \tag{4}$$

where the function $\delta(\texttt{expr})$ returns 1 if `expr` is true, and 0 otherwise. We refer to the quantity $d^{0.5}$ as the *degree of information* of a landscape.

If a search algorithm sampled the search space uniformly, $d^{0.5}$ would correspond to the probability of the algorithm not being forced to make a random decision when comparing a pair of solutions. For example, under the assumption of uniform sampling, a degree of information of $d^{0.5} = 0$ would imply that all the decisions that an algorithm makes are random. Hence, on problems with $d^{0.5} = 0$ (that is, the class of NIAH problems) no algorithm can outperform random search. However, note that a degree of information $d^{0.5} = 1$ does not necessarily imply good performance. A deceptive landscape, for example, may have $d^{0.5} = 1$ and, yet, be very hard, since all the information it contains leads the search algorithm away from the global optimum.

# 4 Performance landscapes

Having restricted our attention to the class of algorithms that use only relative fitness values when making decisions, any performance function for this class of algorithms will be of the form $P = P(X, \mathcal{N}, V, x_{\text{target}})$, where, to reiterate, $X$ is a search space, $\mathcal{N}$ is the neighbourhood structure which is defined over $X$, $V$ is the information landscape and $x_{\text{target}}$ is the target solution (global optimum). In this paper we assume that $X$ and $\mathcal{N}$ are constant, and, so, the performance function can be written as $P = P(V, x_{\text{target}})$ or $P = P(V)$ when $x_{\text{target}}$ is implied. Our objective, in this section, is to build a useful approximation of $P$.

Naturally, for any non-trivial algorithm, $P$ will be a highly complicated function of the $n$ components of $V$, and so we have little hope to derive an explicit formulation for it. However, given a suitable training set, in principle this function could be estimated using machine learning techniques. For a fixed $x_{\text{target}}$, a training set would be made up of pairs of the form $(V_k, P_k)$, $k = 1, 2, \cdots$, where $V_k$ is a particular information landscape – an input for the learner – and $P_k$ is the corresponding performance measure – the target output for the learner. Ideally, we would want $P_k = E[P(V_k)]$.[6] Since we do not know the function $P(V)$, we need to obtain the target values $P_k$ by some other means. These values, for example, can be estimated by averaging the performance recorded by running the algorithm a suitably large number of times over the particular landscape in question.[7]

To illustrate the approach, in this paper we consider the simplest possible approximation for $P$: a linear function of the $n$ variables in $V$ (the entries of the information matrix). That is:

$$P(V) \approx c_0 + \sum_{i=1}^{n} c_i v_i. \tag{5}$$

In order to estimate the coefficients $c_i$ we apply multivariate linear regression over our training set.

[6] The expectation operation is required for stochastic search algorithms. This is because, for such algorithms, the performance function $P(V)$ is a stochastic function and, so, even if we had an explicit formulation for $P(V)$, $P(V_k)$ would take different values for different random seeds.

[7] Naturally, for this to work well, the training set should include a representative set of problems $V_k$. For example, it should include a variety of easy and difficult problems as well as randomly chosen problems.

Naturally, we do not expect that for all algorithms a linear approximation of $P$ will be sufficient to model performance over the whole space of possible information landscapes (although, as will be shown later, surprisingly this appears to be the case at least for simple GAs). However, we expect to be able to find good linear approximations for subsets of problems (information landscapes) with a relatively high degree of similarity. This expectation derives from the following observations.

If $P$ was differentiable, we could use a truncated Taylor expansion in order to approximate $P$ around any reference information landscape $V_0$. That is, for sufficiently small $||V - V_0||$ and a differentiable $P$, we would have

$$P(V) \approx c_0 + \sum c_i(v_i - v_{0_i}) \tag{6}$$

where $c_i = \frac{\partial P(V)}{\partial v_i}\Big|_{V=V_0}$ and $c_0 = P(V_0)$ is the performance of the algorithm on the reference landscape $V_0$. Naturally, even when $P$ is discrete and, therefore, formally non-differentiable we expect it to present some degree of smoothness. So, for a suitably small set of problems not too different from a reference problem $V_0$ we can expect a linear approximation of $P$ to be reasonably accurate. In this case, again, we can use linear regression to find the coefficients in Equation 6. While the reference landscape, $V_0$ can be chosen arbitrary, it is of a particular interest to choose it to be the landscape with *no information* (i.e. $d^{0.5}(V_0) = 0$). This is exactly what we will do in the following, where we will often take as a reference landscape $V_0$ one with elements $v_{0_i} = 0.5$, for $i = 1, \cdots, n$, which we will represent with $\mathbf{0.5}$ hereafter.

Whether using multivariate linear regression or the Taylor expansion around a reference landscape, our approximation to the performance function $P$ depends on the coefficients $c_i$. We denote the vector $C = (c_i)$ as the *performance landscape*.[8] Note that the performance landscape depends on the particular algorithm one wants to study and on the particular measure of performance chosen. Furthermore, for biased algorithms, the performance landscape will, in general, depend also on $x_{\text{target}}$. The information landscape, on the other hand, depends on the fitness landscape alone.

Naturally, the construction of a training set representative of the whole set of possible problems for large landscapes and the corresponding estimation of the performance landscape suffer from the curse of dimensionality. So, we will be able to apply this approach only to study the performance of algorithms on small search spaces. However, the lessons we learn from such small landscapes will be numerous and general. In addition, in the following we will derive techniques to infer properties about problems and algorithms which do not require the full knowledge of the corresponding landscapes.

## 4.1 Quality of information

As indicated in the previous section, the performance landscape is a set of coefficients. Each entry $c_i$ of the performance landscape corresponds to an entry, $v_i$, in the information landscape (see Figure 2) and it represents how the performance of an algorithm is affected by the outcome of the comparison between two particular solutions. The sign of the coefficient $c_i$ tells us which of the two solutions should be chosen if one wants to decide well for the purpose of increasing performance (e.g., reaching the target solution more quickly). The magnitude of the coefficient, $|c_i|$, indicates how big is the expected effect of this choice on the performance of the algorithm.

A simple analysis of Equation 6 reveals that, given the performance landscape $C$ of an algorithm, the information landscape on which such an algorithm should provide best performance, $V_{\max} = (v_{\max_1}, \cdots, v_{\max_n})$, is given by

$$v_{\max_i} = \arg\max_{v_i}[c_i(v_i - v_{0_i})], \tag{7}$$

[8] We define the performance landscape using the coefficient in Equation 6, not those in Equation 5, although these formulations are effectively equivalent as we will show later.

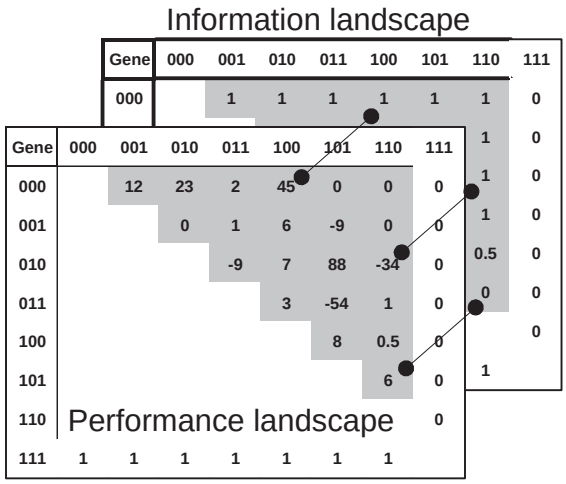



Figure 2: Relation between the information landscape and the performance landscape. Large (positive or negative) values in the performance landscape indicate that the corresponding entries in the information landscape are important. Small values indicate choices that have a limited effect on the performance of the algorithm of interest.

while the information landscape on which the algorithm should provide worst performance, $V_{\min} = (v_{\min_1}, \cdots, v_{\min_n})$, is given by

$$v_{\min_i} = \arg\min_{v_i}[c_i(v_i - v_{0_i})], \tag{8}$$

where each $v_i$ ranges over the set $\{0, 0.5, 1\}$ for information landscapes induced by a deterministic fitness function and $v_i \in [0, 1]$ otherwise. More precisely:

- If $c_i > 0$ then $\max_{v_i}(c_i(v_i - v_{0_i})) = c_i(1 - v_{0_i})$ and this is achieved for $v_i = 1$, while $\min_{v_i}(c_i(v_i - v_{0_i})) = -c_i v_{0_i}$ and this is achieved for $v_i = 0$.
- Vice versa, if $c_i < 0$ then $\max_{v_i}(c_i(v_i - v_{0_i})) = -c_i v_{0_i}$ and this is achieved for $v_i = 0$, while $\min_{v_i}(c_i(v_i - v_{0_i})) = c_i(1 - v_{0_i})$ and this is achieved for $v_i = 1$.
- Finally, if $c_i = 0$ then the entry in the information matrix represented by $v_i$ is not relevant to the search (at least from the point of view of search performance), and so any value of $v_i \in \{0, 0.5, 1\}$ (or $v_i \in [0, 1]$) will provide the same level of performance.[9] We will adopt the convention of setting such elements of $V_{\max}$ and $V_{\min}$ to the value 0.5 (which is effectively a "don't care" symbol, from the algorithm's point of view).

That is, optimal landscapes for a given performance landscape (algorithm) are those where $v_i = 1$ for all $i$ where $c_i > 0$, $v_i = 0$ for all $i$ for which $c_i < 0$, and $v_i$ takes any value for all the remaining $i$'s. The worst possible landscapes are constructed similarly (note $v_{\min_i} = 1 - v_{\max_i}$ for all $i$ where $c_i \neq 0$).

With $V_{\max}$ and $V_{\min}$ in hand, we can now assess the *quality of the information* in a landscape. We do this by using the performance landscape: if the value of an entry in the information landscape matches the value in the optimum information landscape (i.e., $v_i = v_{\max_i}$) then the quality of the information in cell $i$ is good. That is, when faced with the comparison between solutions represented by $v_i$ the algorithm will make the correct decision. If instead, $v_i = v_{\min_i}$, the quality of the information is bad, because the algorithm will make a decision that affects negatively its performance. Naturally, the more entries there are where $v_i = v_{\max_i}$, the better the information in the landscape and, hence, the better the expected decision making of the algorithm. The reverse is true for the entries where $v_i = v_{\min_i}$.

[9] In this case $V_{\max}$ and $V_{\min}$ are entire classes of landscapes, rather than just one landscape.

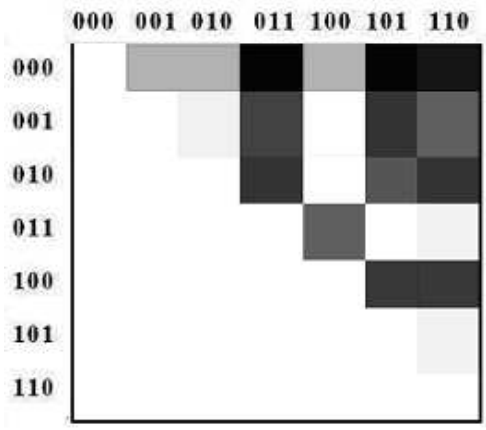



Figure 3: A gray-scale representation of the performance landscape for a simple GA using uniform crossover and no mutation.

The following example illustrates how the coefficients of the performance landscape relate in practice with the quality of information. Let us consider a simple GA using uniform crossover and no mutation, and let us focus on the subclass of problems for which the global optimum is the string 111. Figure 3 shows the performance landscape obtained for such a GA (Section 7.4 provides more technical details on this experiment). In the figure, the darker a square, the larger the magnitude of the corresponding entry in the performance landscape.

A careful analysis of the figure reveals the relationship between the quality of the information and the performance landscape. For example, let us focus on the entry in the first row and last column, which represents the impact on performance of the comparison between the solutions 110 and 000. This is quite dark, signifying that a problem whose information landscape, deceptively, induced the GA to choose 000 over 110, would prove to be harder than a problem where the opposite is true. This makes sense since, in a GA, comparisons between strings are used in the selection phase to decide which individuals to use to continue the search and, under uniform crossover, on average, the chance of obtaining the global optimum (the string 111) using 110 is much higher than using the string 000. As another example, let us concentrate on the entry representing a comparison between 110 and 011. This is rather light, indicating that any decisions involving this comparison have a limited impact on performance. Why? Because, under uniform crossover, the chance of reaching 111 from 011 rather than 110 is, on average, approximately the same.

This example illustrates that the notion of performance landscape captures important features of an algorithm, as far as performance is concerned, despite being based on a linear approximation. More evidence for this will be provided in the following sections.

## 4.2 Performance vectors

We have seen how the information landscape may be computed starting from a fitness function and, how, conversely, it is possible to reconstruct a representative fitness function, $f_{\text{rank}}$, from an information landscape. Clearly, the fitness function is a more concise representation, being of size $O(|X|)$ instead of $O(|X|^2)$.

The performance landscape, like the information landscape, is of size $O(|X|^2)$ and, so, one might wonder whether a coarser-grained representation for the quality of information from the point of view of an algorithm might exist. Taking inspiration from the definition of $f_{\text{rank}}$ (Equation 2) we can create row-by-row summaries of the performance landscape. In particular, if we denote the entries of the performance landscape matrix with $c_{i,j}$, we propose the following two

quantities, which we call *performance vectors*:

$$c_{\text{rank}}(i) = \sum_j \delta(c_{i,j} > 0) \tag{9}$$

and

$$c_{\text{sum}}(i) = \sum_j c_{i,j}. \tag{10}$$

The rank-based performance vector $c_{\text{rank}}(i)$ counts in how many cases solution $i$ is more important than another solution in the search space for the purpose of maximising the performance of the algorithm. This gives us a rough estimate of the relative importance of sampling different solutions in the search space. The sum-based performance vector $c_{\text{sum}}(i)$ provides a finer grain importance measure, which considers also by how much a solution is more (or less) important than another.

The proximity (w.r.t. the search operators) of solutions with high fitness is a fundamental underlying assumption of modern search heuristics. When considering optimisation problems with one global optimum, this assumption is even stronger – high fitness is expected to indicate proximity to one, single solution – the global optimum. This assumption is reflected in the expected values of $c_{\text{rank}}(i)$ or $c_{\text{sum}}(i)$ w.r.t. the proximity of $x_i$ to $x_{\text{target}}$. High values of $c_{\text{rank}}(i)$ or $c_{\text{sum}}(i)$ indicate that $x_i$ is relatively close (w.r.t. the search operators) to $x_{\text{target}}$. We will see this explicitly in Section 7.4.2.

# 5 Properties of search algorithms in the black-box scenario

As we mentioned in Section 1, search in the black box scenario is not yet fully understood. Different theoretical approaches have explored the nature of search from different perspectives, but none has provided a full account. In this section we discuss how the information landscape framework can contribute to the understanding of search in the black box scenario. In particular, Section 5.1 will give a geometrical representation of the relation between a search algorithm and a particular problem. Section 5.2 will use our linear approximation of the performance function $P(V)$ to compute the expected performance of algorithms and explore its relationship with the NFLTs. Section 5.3 will discuss how the framework can help distinguish between different notions of randomness in search. Finally, Section 5.4 will introduce and characterise the notion of negation of a problem, showing that those structural properties of a problem that contribute to the good performance of a particular algorithm cause the same algorithm to perform badly on the negated problem.

## 5.1 Geometric interpretation

Each problem (information landscape) contains a certain amount of information which is explicitly quantified by the degree of information. This is defined irrespectively of the search algorithm being used. The quality of the information, instead, is defined according to the specific search algorithm under consideration. In our framework, this is represented by the performance landscape. This separation of roles leads to a new perspective in which search-algorithms in general can be viewed.

Equation 6 can be rewritten as

$$\begin{aligned} P(V) &\approx c_0 + \sum_i c_i (v_i - v_{0_i}) \\ &= \sum_i c_i v_i + c_0 - \sum_i c_i \, v_{0_i} \\ &= C \cdot V + \tilde{c}_0 \end{aligned} \tag{11}$$

where $\tilde{c}_0 = c_0 - \sum_i c_i \, v_{0_i}$ is a constant and "$\cdot$" represents the scalar product. So, up to a constant, *the performance of a search algorithm on a problem can approximately be interpreted as the scalar*

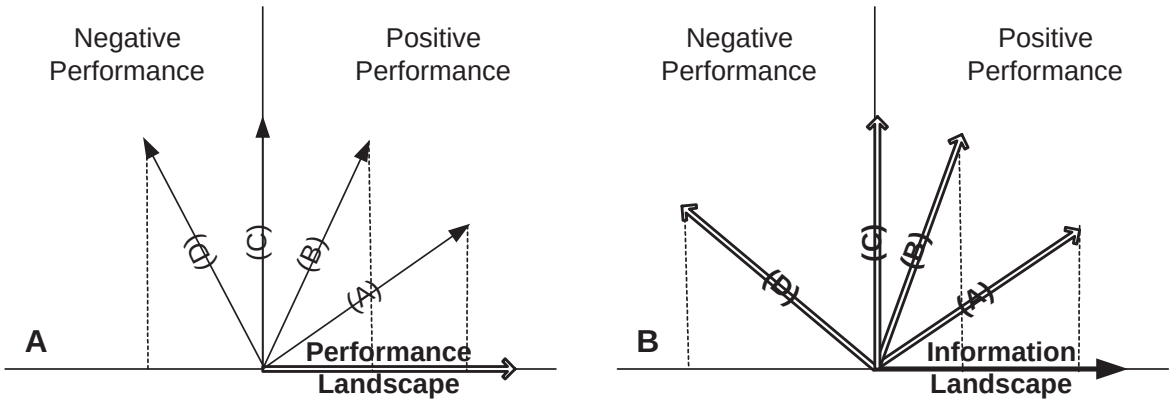



Figure 4: (A) Comparing problems: the performance landscape for a particular algorithm (thick horizontal arrow) and four information landscapes (problems) of increasing difficulty (thin arrows labelled A–D). (B) Comparing algorithms: the information landscape for a particular problem (thin horizontal arrow) and four performance landscapes (search algorithms) of increasing inefficiency (thick arrows labelled A–D).

*product between the corresponding information and performance landscapes.* The former represents the search problem, the latter the performance characteristics of the search algorithm.

The scalar product interpretation can help us visualise geometrically the connection between the information landscape and the performance landscape. To our first order approximation and ignoring the constant offset $\tilde{c}_0$, the performance of an algorithm is simply the magnitude (with sign) of the *projection* of a vector $V$ (representing the information of a problem) onto a vector $C$ (representing the sensitivity of the search algorithm to that information). Figure 4 shows how this can be used to compare the performance on different problems of a particular search algorithm as well as to compare the performance of different search algorithms on a particular problem.

Note that the NFLTs suggest a similar, but not identical, interpretation (Wolpert and Macready, 1997), predicting that the performance of an algorithm is the scalar product between a vector representing a probability distribution over all possible fitness functions and a vector representing the probability of obtaining a particular sample of solutions within a certain number of attempts for each possible fitness function. The relationship between the NFLTs and our framework is further discussed in the next sections.

## 5.2 Expected performance of search algorithms

Let us compute the expected performance for an algorithm. Naturally, in order to do so, we need to define a probability distribution over the space of possible fitness functions $\Pr(f)$ (the uniform probability distribution assumed in the original NFLT being just one possibility). Since in the information landscape framework we restrict our attention to algorithms that consider only relative fitness, any fitness function $f$ can equivalently be represented with a corresponding information-landscape/target-solution pair $(V, x_{\text{target}})$. The space of all possible problems can be, therefore, represented as the Cartesian product $\mathcal{V} \times X$ where $\mathcal{V}$ is the space of all valid information landscapes. The distribution $\Pr(f)$ is equivalent to $\Pr(V, x_{\text{target}})$ over $\mathcal{V} \times X$.

Let us compute the expected performance for an algorithm:

$$\begin{aligned}
E[P] &= \sum_{V\in\mathcal{V}}\sum_{x\in X}\Pr(V,x)P(V,x)\\
&\approx \sum_{V\in\mathcal{V}}\sum_{x\in X}\Pr(V,x)\left(C(x)\cdot(V-V_0)+c_0(x)\right)\\
&= \sum_{x\in X}c_0(x)\overbrace{\sum_{V\in\mathcal{V}}\Pr(V,x)}^{=\Pr(x)}+\sum_{x\in X}C(x)\cdot\overbrace{\sum_{V\in\mathcal{V}}\Pr(V,x)(V-V_0)}^{\Pr(x)\sum_V\Pr(V|x)(V-V_0)}\\
&= \sum_{x\in X}\Pr(x)c_0(x)+\sum_{x\in X}\Pr(x)C(x)\cdot(E[V|x]-V_0)\\
&= E[c_0(x)]+E\big[C(x)\cdot(E[V|x]-V_0)\big]
\end{aligned}$$

where in the second line of the derivation we used our linear approximation $P(V,x)\approx C(x)\cdot(V-V_0)+c_0(x)$, $C(x)$ being the performance landscape of the search algorithm corresponding to the target solution $x$ and $c_0(x)$ being the corresponding constant term.

If the distribution over $\mathcal{V}$ is chosen to be independent from the distribution of $x_{\text{target}}$, that is, $\Pr(V,x)=\Pr(V)\Pr(x)$ and $\Pr(V|x)=\Pr(V)$, then $E[V|x]=E[V]$, whereby

$$E[P]\approx E[c_0(x)]+E[C(x)]\cdot(E[V]-V_0). \tag{12}$$

For unbiased search algorithms, if $V$ and $x_{\text{target}}$ are independent, we should not expect the average performance to depend on $x_{\text{target}}$. That is, we should expect $E[c_0(x)]=c_0(x_{\text{any}})$ and $E[C(x)]=C(x_{\text{any}})$, where $x_{\text{any}}$ is any particular point in $X$ we may want to choose as our global optimum. So, for this important class of algorithms we have

$$E[P]\approx c_0(x_{\text{any}})+C(x_{\text{any}})\cdot(E[V]-V_0). \tag{13}$$

It is easy to show that we obtain exactly the same approximation for $E[P|x_{\text{target}}=x_{\text{any}}]$. This means that we can infer the behaviour of this class of algorithms on all possible problems simply by studying and extrapolating their behaviour on the subset of problems with the same global optimum $x_{\text{target}}$. These problems are represented by a distribution $\Pr(V,x)=\Pr(V)\Pr(x)$ with $\Pr(x)=\delta(x=x_{\text{target}})$. We will effectively do this from Section 5.3 onward.

By definition, from our linear approximation we obtain

$$P(E[V],x_{\text{any}})\approx c_0(x_{\text{any}})+C(x_{\text{any}})\cdot(E[V]-V_0) \tag{14}$$

for any type of algorithm and problem distribution. If, however, we restrict ourselves to the class of unbiased search algorithms and independent $V$ and $x_{\text{target}}$ we can link the previous two results obtaining

$$E[P(V,x_{\text{any}})]\approx P(E[V],x_{\text{any}}). \tag{15}$$

Before we continue with our analysis, let us try to understand the meaning of Equation 15 and its relationships with NFLT results.

We begin with the term $P(E[V],x_{\text{any}})$, i.e., the performance of an algorithm on the expected information landscape. If we limit the elements of the information landscapes to be in the set $\{0,0.5,1\}$ then, depending on the characteristics of $\Pr(V)$, $E[V]$ may or may not be a member of the space of information landscapes $\mathcal{V}$. If $E[V]\in\mathcal{V}$ the semantics of $P(E[V],x_{\text{any}})$ is clear. However, in general, the elements of $E[V]$ can be any real numbers in the interval $[0,1]$ and, so, $E[V]\notin\mathcal{V}$. In this case we can interpret $E[V]\notin\mathcal{V}$ as the landscape induced by a noisy fitness function (see Section 3), so we can still run our algorithm on such a landscape and measure the performance function $P(E[V],x_{\text{any}})$.

Interestingly, if we consider a set of functions for which the NFLTs hold (*NFL set*), we have $\Pr(V)=\frac{1}{|\mathcal{V}|}$ and $E[V]=\mathbf{0.5}\in\mathcal{V}$ as one can easily verify. This holds also for the sharpened NFLT

(Schumacher et al., 2001) and the non-uniform sharpened NFLT (Igel and Toussaint, 2004). In these cases, the pair $(E[V], x_{\text{target}})$ represents a NIAH landscape. So, Equation 15 predicts that the average performance of any unbiased search algorithm over a *NFL set* is (approximately) the same as the performance of the algorithm on the NIAH.

This prediction turns out to be exact, at least for the case where performance is measured as the number of fitness evaluations required to sample the global optimum. For this performance measure, (Droste et al., 2003) have shown that the performance of any search algorithm on the NIAH is $\frac{|X|+1}{2}$. On the other hand, (Igel and Toussaint, 2004) have shown that the expected performance of any, non-resampling algorithm on a NFL set is $\frac{|X|+1}{n+1}$, where $n$ is the number of target solutions. Clearly, for $n = 1$ (which we assume in Equation 15) the performance of an algorithm on the NIAH is the same as the expected performance of the algorithm on a NFL set, i.e., $E[P(V, x_{\text{any}})] = P(E[V], x_{\text{any}})$. So, in these conditions, Equation 15 is exact.

Continuing our analysis, if, in addition to the statistical independence of landscape and optimum, we choose $V_0 = E[V]$, then from Equation 12 it follows that

$$E[P] \approx E[c_0(x)] \tag{16}$$

for generic algorithms and from Equation 13 it follows that

$$E[P] \approx c_0(x_{\text{any}}) \tag{17}$$

for unbiased search algorithms. Note that $c_0(x_{\text{any}})$ is a constant that represents the performance value estimated using our linear approximation to $P$ for $V = E[V]$, so we expect $c_0(x_{\text{any}})$ to be very close to the *actual* performance on $E[V]$, $P(E[V], x_{\text{any}})$.

Let us now restrict our attention to algorithms that do not resample and let us choose a distribution of problem where $\Pr(V, x)$ factorises into $\Pr(x) = \frac{1}{|X|}$ and $\Pr(V) = \frac{1}{|\mathcal{V}|}$. Because of the NFLTs, in these conditions we expect $E[P]$ to be independent of the algorithm, and, via Equation 16, also $E[c_0(x)]$ to be algorithm independent. Furthermore, from Equation 17, for unbiased search algorithms we expect $c_0(x_{\text{any}})$ to be algorithm independent (in addition to being independent from $x_{\text{any}}$, of course).

Random search is an unbiased algorithm. So, a non-resampling version of random search must show the same performance on any choice of $V$ and $x_{\text{target}}$. This means that $C(x_{\text{any}}) = \mathbf{0}$ (a null vector) irrespective of $\Pr(V, x)$. So, $E[P] \approx c_0$ which is also the performance of (non-resampling) random search. This is the interpretation of $c_0$ we will use in the following section. Naturally, if performance is measured as the number of fitness evaluations required to visit the global optimum, then we can use the connection between NIAH and expected performance established earlier to provide a more precise value for $c_0$, namely

$$c_0 = \frac{|X| + 1}{2}.$$

## 5.3 Search randomness

Under the standard NFLT assumptions (see Section 5.2) and taking $V_0 = \mathbf{0.5}$, we can see that the scalar product $C \cdot (V - V_0)$ in Equation 11 measures by how much the performance of an algorithm on a particular problem differs from the performance of random search ($c_0$). A positive scalar product corresponds to better performance than random search, while a negative scalar product corresponds to worse performance than random search.

Taking $V_0$ as the origin of problem space, an algorithm cannot do better nor worse than random search when the vector representing the problem, $V - V_0$, is *orthogonal* to the vector representing the algorithm, $C$, and *vice versa*. This situation is represented in Figure 4, where the vectors labelled with C are orthogonal to the horizontal vectors representing a performance landscape (Figure 4.A) and an information landscape (Figure 4.B). Note that near-orthogonality of $C$ and $V - V_0$ should be expected whenever no attempt has been made to match problem and algorithm. In this situation the entries of $V - V_0$ will be uncorrelated to the entries of $C$ and, since the

elements of both vectors can be either positive or negative, we should expect $C \cdot (V - V_0) \approx 0$ due to cancellations.

As mentioned before, there are two other scenarios in which the performance of an algorithm will be $c_0$. Irrespectively of the algorithm, if $V = V_0$ (that is the case of the NIAH where the degree of information equals zero) the performance will be equal to that of random search. Similarly, whenever $C = \mathbf{0}$, irrespectively of the particular landscape, the performance will be equal to that of random search. In this case, since all $c_i$'s are zero, the algorithm completely ignores the fitness function. That is, it does some sort of exhaustive *blind search* (e.g., enumeration).

Note that while we can easily infer in a variety of ways what the performance in the two scenarios $V = V_0$ and $C = \mathbf{0}$ is, it is in general not clear for which problems and algorithms one should still expect an algorithm to perform like random search. Our framework makes it clear that the space of problems where this happens is a plane in $\mathbb{R}^n$ orthogonal to the performance landscape $C$. In addition we are able to evaluate performance in other (generally more interesting and realistic) scenarios, where, for example, the performance of an algorithm is somehow better than that of random search.

To summarise, the theory predicts that performance will be identical in the following four scenarios:

1. The algorithm performs a blind exhaustive search which ignores the information in the landscape (i.e., $C = \mathbf{0}$).

2. The algorithm searches explicitly in a random way (random search), irrespective of the information landscape (again, $C = \mathbf{0}$).

3. The algorithm does not search in a random way, but there is no information in the landscape to guide the search (i.e., $V = \mathbf{0.5}$, the problem is a NIAH).

4. The algorithm does not search in a random way but the information given by the landscape is random (i.e., $C \perp V$).

This is, of course, valid under the assumption that no resampling takes place. In practice, however, the vast majority of search algorithms cannot avoid resampling. In this case blind search and the three different types of random search described above may lead to different levels of performance. This is true because each of the four has a different sensitivity to resampling.

In the first algorithm, blind exhaustive search, it is often possible to avoid resampling (e.g., enumeration of the search space of fixed-length strings can simply be performed with a suitable number of nested `for` loops).

The second algorithm, random search, similarly to the previous one, does not assume anything about the structure of the landscape. Here, however, resampling is typically hard to avoid. This is because, in a truly random search, the only way one could avoid resampling is by storing all solutions that have been already visited, or, all solutions that still need to be visited, which would require huge amounts of memory even for small problems.

In the third case, i.e., when there is no information in the landscape, one might think that the performance of an algorithm should be equivalent to that of random search, because all decisions of the algorithm are now random. However, this is not the case because, typically, the search operators of the algorithm tend to focus on areas close to the current fit solutions. The fact that they focus on small areas makes them more prone to resampling (Borenstein and Poli, 2004).

The fourth case is only slightly different from the third. The difference between the two is that in one case the outcome of every comparison (an entry in the information landscape) is randomly fixed, whereas in the other case it is random. This means that, if the algorithm does not resample, a random landscape and a landscape with no information (i.e., a NIAH) should present the same level of difficulty. With resampling, however, an algorithm searching a landscape with no information may make different decisions when faced with the same comparison multiple times, whereas the same algorithm searching a random landscape will always make the same decision. The slight additional randomness induced by landscapes with no information helps preserve diversity in

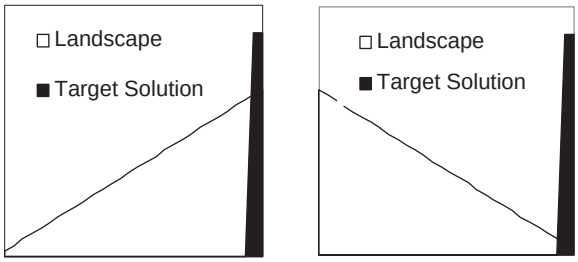



Figure 5: The effect of negating a landscape. On the left the target solution combines with a unimodal landscape to create an easy unimodal function. On the right, the negation of the landscape is effectively a deceptive problem.

population-based search algorithms. As a result, performance may be better for NIAH landscapes than for random landscapes.

## 5.4 Negation of problems and performance symmetries

In order to gain insights about a particular algorithm, researchers have tended to study the behaviour of the algorithm either on very easy problems or on very difficult ones. In the GA community, for example, the study of easy problems, such as Onemax, on the one hand, and very difficult problems, such as the class of trap functions, on the other hand, has been extensive.

From the information landscape perspective, a deceptive problem is difficult because it contains misleading information, in the sense that the information in the problem is negatively correlated with the algorithm's optimum seeking strategy. On the contrary, an easy problem, such as Onemax, is one where the information in the landscape is positively correlated with the behaviour of the algorithm. This suggests that although, traditionally, the classes of easy and hard problems have been typically studied separately, there may be a strong, unsuspected, connection between them. Indeed, as we will show below, easy and deceptive problems are just a reflection of one another, or, as we put it, the negation of one another.

We define the *negation* of an information landscape $V$ as:

$$\overline{V} \equiv \mathbf{1} - V \tag{18}$$

where $\mathbf{1} = (1, \cdots, 1)$ is an information landscape with all entries equal to 1. That is, the negation of an information landscape is a landscape where all the comparisons between pairs of elements of the reduced search space $X' = X \setminus \{x_{\text{target}}\}$ give the opposite outcomes to those of the original landscape. The elements of the original landscape that had value 0.5 remain unchanged. Note that the target solution (global optimum) is not part of the landscape, and, so, it is the same for original and negated landscapes. To give an impression of the relationship between a landscape and its negation, in Figure 5 we provide an illustration of the effects of negating a simple *fitness* landscape. Clearly, negation changes the easy unimodal landscape on the left (where all the information in the landscape points towards the global optimum) into the hard deceptive landscape on the right (where all the information points away from the global optimum).

More formally we can show that the performance of an algorithm on an information landscape $V$ is related to the performance of the same algorithm on the negated information landscape $\overline{V}$ by the following approximation (where we omit the dependency of $P$ on $x_{\text{target}}$):

$$P(\overline{V}) + P(V) \approx 2c_0. \tag{19}$$

Proving this result is very easy. From Equation 11 we have

$$\begin{aligned} P(\overline{V}) + P(V) &\approx C \cdot (\overline{V} - V_0) + c_0 + C \cdot (V - V_0) + c_0 \\ &= C \cdot (\mathbf{1} - V - V_0) + C \cdot (V - V_0) + 2c_0 \\ &= C \cdot (\mathbf{1} - 2V_0) + 2c_0 \\ &= 2c_0 \end{aligned}$$

where in the last step we used the definition $V_0 = \mathbf{0.5}$.

That is, for algorithms where no resampling takes place, if an algorithm performs better than random search by a certain amount on a problem, the same algorithm will perform worse than random search and by exactly the same amount on the negation of this problem, and *vice versa*. This implies a *symmetry with respect to random search*. For algorithms that resample, Equation 19 is still valid, but $c_0$ should be interpreted as $P(V_0)$. So, in this case $P(V)$ and $P(\overline{V})$ are *symmetric with respect to the performance of the algorithm on the NIAH problem*.

Geometrically, we can interpret this result very easily. If we place the origin of problem space at $V_0$, the negation $\overline{V}$ of an information landscape $V$ is a vector symmetric to $V$ with respect to the origin. So, if, for example, a problem is maximally aligned with the performance landscape of an algorithm, thereby providing best performance, the negation of the problem will be maximally misaligned with it, producing the worst possible performance.

Figure 4.A illustrates the situation. The information landscape represented by vector B is the negation of the information landscape represented by D. The magnitude of the projections of the two vectors on the performance landscape is the same, but the sign is different. Note the symmetry of these vectors with respect to the information landscape represented by vector C, which is orthogonal to the performance landscape of the algorithm.

The negation of a landscape that is orthogonal to the performance landscape of an algorithm is still orthogonal to it. So, an information landscape uncorrelated with the performance landscape, e.g., a random landscape, will still be uncorrelated upon negation.

A consequence of the symmetry with respect to $c_0$, irrespective of whether this is interpreted as the performance of random search or the performance of the given algorithm on a NIAH problem, is that the better the performance of an algorithm is on one landscape the worse it will be on another one. This clearly provides an additional sanity check of our results with respect to the NFLTs.

Naturally, one could also easily define the negation of an algorithm – a concept that leads to other interesting symmetries – but we will explore this in future research.

# 6 A predictive measure of problem difficulty

In the previous section we focused on the general implications of the information landscape framework for search in the black box scenario. We did not consider any specific search algorithm. However, unlike other theoretical approaches, it is easy to apply this framework to particular search algorithms. In theory all we need to do is to estimate the performance landscape of an algorithm by doing a linear regression on a dataset. Once the performance landscape is available, it is then possible to estimate the approximate difficulty of any problem via a simple scalar product.

We will follow this approach in Section 7.1. Unfortunately, due to the sheer size of the spaces involved, calculating a performance landscape is a computationally expensive process. This prevents the estimation of performance landscapes for realistic problem sizes. In this section, we will attempt to remove this obstacle and see to what extent the information landscape alone can be used as an estimator of problem hardness.

In Section 4.1 we argued that the quality of the information in a landscape, from the point of view of a particular algorithm, can be evaluated by comparing each entry in the information landscape with the corresponding entry in the performance landscape. In particular, an entry $v_i$ provides positive information (improves the performance) if either $v_i = 1$ and the corresponding entry in the performance landscape, $c_i$, is positive or $v_i = 0$ and $c_i < 0$. The entry $v_i$ provides

negative information (which decreases performance) if either $v_i = 1$ and $c_i < 0$ or $v_i = 0$ and $c_i > 0$. So, to assess quality of information all we need to know is the *signs of the elements in the performance landscape*. Naturally, the performance of an algorithm is expected to improve as the number of landscape elements providing positive information grows. In theory, using the vector $C$, we could be more precise.[10] However, as we already mentioned we do not want to use any direct information of the performance landscape since this is hard to compute. So, what can we do in this situation?

If we could somehow determine the signs, $c_i/|c_i|$, of the elements $c_i$ of the performance landscape without actually computing the $c_i$'s, we could use these to determine the proportion of positive- vs. negative-information-providing entries in an information landscape. As explained above, this would be enough to give us a rough estimate of how hard the landscape would be for a particular search algorithm. So, to make progress we need an approximation for this sign matrix.

In Section 4.1, on the assumption that our first order approximation to the performance function $P(V)$ is reasonably accurate, we predicted the easiest and hardest information landscapes for a search algorithm. These are the landscapes $V_{\max}$ and $V_{\min}$ given in Equations 7 and 8, respectively. If $V_{\max}$ is available, this effectively gives us the signs of the elements of $C$. So, given a landscape $V$, *the number of non-matching entries between $V$ and $V_{\max}$ is a rough indicator of problem difficulty*. Naturally, $V_{\max}$ is a function of the performance landscape $C$, which we do not have, so we would appear to be back to the original problem. This is not the case, however, because we can estimate $V_{\max}$ empirically. In fact, what we are looking for is a really easy landscape $V_{\text{easy}}$ – the easier the merrier – which we can use as an estimate for $V_{\max}$.

More formally, we propose to use as an *indicator of problem difficulty* the quantity

$$h(V) = d(V, V_{\text{easy}}), \tag{20}$$

where $h$ is mnemonic for "hardness" and $d(\cdot, \cdot)$ is a distance measure between landscapes. If $V_1 = (v_{1_1}, v_{1_2}, \cdots, v_{1_n})$ and $V_2 = (v_{2_1}, v_{2_2}, \cdots, v_{2_n})$ are two information landscapes, we define the distance between them as

$$d(V_1, V_2) = \frac{1}{n} \sum_i |v_{1_i} - v_{2_i}|. \tag{21}$$

For landscapes with full information (i.e., without any 0.5 entries), the distance between two landscapes is the proportion of non-matching entries in the two vectors representing the landscapes. So, in these conditions $d$ is the Hamming distance.[11]

To reiterate, the function $h(V)$ is not an exact measure of hardness because it does not consider the fact that different entries in the performance landscape may have very different values, since some choices are more important than others. However, we expect $h(V)$ to be a reasonable approximation.

We can further increase our confidence in this choice with some simple calculations. For example, it is possible to approximately relate the distance between two landscapes $V_1$ and $V_2$, and the difference in performance recorded for them, as follows:

$$\begin{aligned} |P(V_1) - P(V_2)| &\approx |C \cdot (V_1 - V_0) + c_0 - C \cdot (V_2 - V_0) - c_0| \\ &= |C \cdot (V_1 - V_2)| \\ &\leq ||C|| \times ||V_1 - V_2||, \end{aligned} \tag{22}$$

where in the last step we used the Cauchy-Schwarz inequality and $||\cdot||$ is Euclidian norm. If the degree of information in $V_1$ and $V_2$ is 1, it is trivial to show that the Euclidean distance $||V_1 - V_2||$ between the landscapes is also identical to $\sqrt{n \times d(V_1, V_2)}$. If we then take one of the two landscapes in Equation 22 to be a known reference landscape, then the previous expression gives us upper and lower bounds for the performance of the algorithm. Particularly interesting

[10] The extent to which each entry $v_i$ contributes to performance is approximated by the *magnitude* of the corresponding entry in the performance landscape $c_i$.

[11] Naturally, any other reasonable distance measure would have served the purpose. We chose Equation 21 for its simplicity and for its connection to the Hamming distance.

is the case in which the reference landscape is $V_{\max}$, because we know that the corresponding $P(V_{\max})$ is the best possible. In this case we obtain the bound:

$$P(V_{\max}) - ||C|| \times ||V_{\max} - V|| \leq P(V) \leq P(V_{\max}), \tag{23}$$

which, for landscape of degree 1 and assuming $V_{\max} \approx V_{\text{easy}}$, we can rewrite as

$$P(V_{\max}) - \alpha\sqrt{h(V)} \leq P(V) \leq P(V_{\max}) \tag{24}$$

where $\alpha = \sqrt{n}\,||C||$ is a constant. This shows how our measure of problem difficulty naturally emerges from the information landscape framework.

An even more convincing result, in this sense, is provided by the following alternative bound:

$$\begin{aligned}
|P(V_{\max}) - P(V)| &\approx \left|\sum_i c_i(v_{\max_i} - v_i)\right| \\
&\quad \text{(by definition of } V_{\max}, \forall i\, c_i(v_{\max_i} - v_i) \geq 0) \\
&= \sum_i |c_i| \times |v_{\max_i} - v_i| \\
&\leq \left(\max_i |c_i|\right) \sum_i |v_{\max_i} - v_i| \\
&\approx \left(\max_i |c_i|\right) \sum_i |v_{\text{easy}_i} - v_i|,
\end{aligned}$$

whereby

$$P(V_{\max}) - \beta_{\max} \times h(V) \leq P(V),$$

where $\beta_{\max} = (\max_i |c_i|) \times n$ is a constant. Similarly, it is possible to show that

$$P(V_{\max}) - \beta_{\min} \times h(V) \geq P(V),$$

where $\beta_{\min} = (\min_i |c_i|) \times n$. So, $h(V)$ gives us the following approximate upper and lower bounds for $P(V)$:

$$P(V_{\max}) - \beta_{\max} \times h(V) \leq P(V) \leq P(V_{\max}) - \beta_{\min} \times h(V). \tag{25}$$

This result is applicable also to landscapes with degree of information less than 1.[12]

Equations 23, 24 and 25 confirm the intuition behind our measure of problem difficulty. That is, for a fixed algorithm, the further a landscape is from the optimum landscape, the wider the decrease in performance the algorithm may exhibit on it.

# 7 Empirical evaluation

It is hard to assess mathematically the accuracy of our framework. This is because the framework is applicable to search algorithms in general, but, potentially, each algorithm has a different performance function $P(V)$. In addition, it is exceptionally hard to build an explicit formulation for this function, even when considering a specific search algorithm (e.g., a GA). So, empirical validation is the only viable strategy.

Focusing on GAs, in this section, we describe empirical evidence which strongly corroborates the framework. In particular, we show that the performance landscape can be used in order to predict the performance of the algorithm on unseen problems. In Section 7.1, we conduct an exhaustive analysis on small landscapes (for the search space of binary strings of length 3) and show that this is indeed the case. Then, moving away from landscapes of a small size towards more realistic sizes (14 bits), we use various examples of known problems from the literature (e.g.,

[12] For some algorithms, $\beta_{\min} = 0$, and so the upper bound in Equation 25 is simply $P(V_{\max})$. For random search and other blind search algorithms also $\beta_{\max} = 0$, in which case $P(V) = P(V_{\max})$.

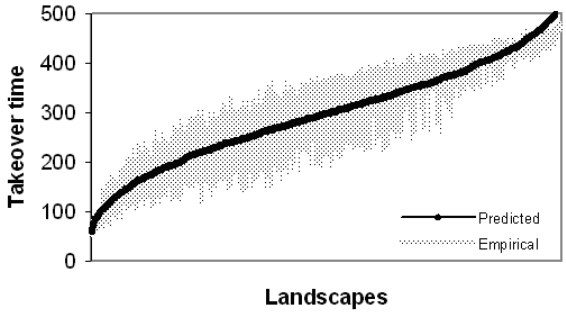



Figure 6: The predicted performance against the empirical performance on the training set (including all possible valid landscape of size 8).

multi modal landscapes, NIAH, MAXSAT, etc.) in order to show that the hardness of a problem can be estimated by measuring its distance from a easy reference landscape using Equation 20 (Section 7.2). In Section 7.3, we demonstrate that, as predicted in Section 5.4, there is a symmetry in the performance of landscapes and their negations. Finally, in Section 7.4, we explore other ways in which the information landscape framework can be used to understand the properties of specific search algorithms.

## 7.1 Exhaustive analysis

In this section we test our main hypothesis and we show that our framework can be used in order to estimate the performance of a GA. Since this requires a full knowledge of the performance landscape, we provide results for small landscapes.

We used a simple GA with one-point crossover applied with 100% probability. The takeover time (i.e., the time required for the entire population to converge to the target solution) was used as the performance measure.[13] We used a population size of 14. The maximum number of generations was 500. The search on each landscape was repeated 1000 times so as to obtain accurate averages. The target solution (global optimum) was excluded from the first generation.

In a first experiment, we measured the mean takeover time for *all possible valid landscapes* of degree 1. These are landscapes which can be derived from a fitness function (see Section 3) and where none of the elements is 0.5 (see Equation 4). It is important to emphasise that the target solution was fixed. Therefore, we were measuring the performance of a GA on all possible landscapes given that the optimum is at a particular position in the search space. Since the size of the search space $X$ is 8, the reduced search space $X'$ is of size 7 and, so, we have $7! = 5040$ possible landscapes. In order to estimate the performance landscape (Equation 6) we used multivariate linear regression on the results obtained from running the GA over all such landscapes. Figure 6 shows a plot of the predicted performance for each landscape (solid line) against the empirical results. For easier visualisation landscapes were ordered on the basis of their predicted performance (from best to worst). The correlation coefficient between observed and predicted performance is 0.935, which indicates that a linear approximation to the performance function is actually rather accurate.

In order to verify whether the linear approximation to $P(V)$ generalises well, we sampled 1000 additional landscapes out of the entire space of possible information landscapes (i.e., including invalid landscapes, see Section 3). Figure 7 shows plots of the predicted performance (using the performance landscape previously calculated) against the empirical results for the new landscapes. The correlation between prediction and observation is still very high (0.923), suggesting good generalisation.

[13] In our framework we have always used the convention that the bigger $P(V)$, the better the performance. However, symmetric results could be derived for the case where the lower the values of $P(V)$ the better. In the present context, the takeover time is used in the latter sense, since we are assuming that the algorithm is stopped

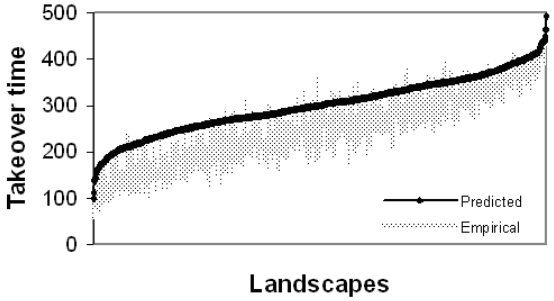



Figure 7: The predicted performance against the empirical performance on a test set of 1000 random valid and invalid landscapes.

## 7.2 Estimation of problem hardness

In the previous section we have shown that our framework can be used in order to accurately estimate the performance of a GA and assess problem difficulty. However, it is clear that a direct estimation of the performance landscape can only be used for small search spaces. This limits the applicability of the approach to theoretical studies. In this section, rather than using the performance landscape directly, we use the ideas presented in Section 6 to see if the approach can be applied to more practical scenarios.

In Section 6 we argued that the hardness of a problem can be estimated using the distance of its information landscape from the optimal landscape $V_{\max}$. Since, in general, the optimal landscape is not known, we proposed to use an approximation, $V_{\text{easy}}$, instead (Equation 20). The question now is which easy problem to choose. We know from many empirical studies that unimodal problems tend to be GA-easy, the Onemax problem being a glowing example. The Onemax belongs to the following more general class of functions: $f(x) = \sum_i \delta(x_i = x_{\text{target}_i})$ where $x_{\text{target}}$ is the global optimum. Onemax, is a specific case, where $x_{\text{target}}$ is the string of all ones. In this work we decided to use the information landscape $V_{\text{easy}}$ derived from $f(x)$ as an approximation of the optimal landscape $V_{\max}$.

The information landscape and the performance function are defined for a fixed target solution. If we change the target solution the same information landscape can change from being easy to being difficult (e.g., consider the information landscape induced by the Onemax function where we change the optimum to being the string 000). So, the distance between landscapes must be computed for landscapes with the same global optimum. This requires knowing the global optimum in advance.

In the experiments reported in this section, we measured the distance between the actual landscape induced by a problem and the one induced by the function $f(x)$ using the global optimum of the problem as $x_{\text{target}}$. We used a simple GA with uniform crossover applied with 100% probability and mutation applied with 10% probability. The search space included binary strings of 14 bits. We used a population size of 20. The first generation in which the optimum was found was used as the performance measure. The results are averages of 100 runs.

The remainder of this section is organised as follows. In Section 7.2.1, empirical results are given for various problems. In Section 7.2.2 we test our approach on three counterexamples for other measures of problem difficulty. Finally, Section 7.2.3 suggests a way in which our measure of problem difficulty can be further improved.

### 7.2.1 Hardness of standard test problems

In this subsection we estimate the hardness of problems with no information (NIAH), random information or random problems (RAND), maximally reliable information (unimodal functions) and maximally unreliable information (deceptive functions). Furthermore, we study problems with

when a population is filled up with copies of the global optimum. So, the sooner this happens the better.

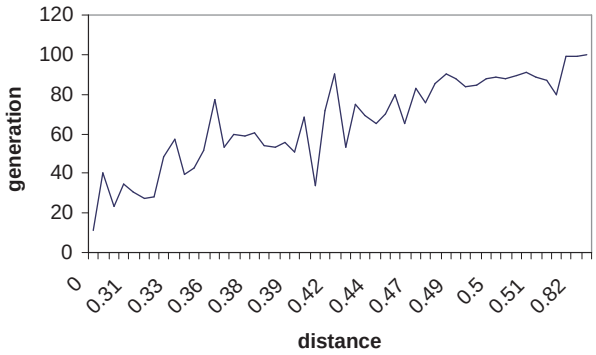



Figure 8: The performance of a GA vs. the predicted difficulty $h(V)$ (the distance from an easy unimodal landscape).

Table 1: Estimated hardness $h(V)$ of our test problems.

| $h(V)$ | Problem |
|---|---|
| 0.00-0.20 | MM1 |
| 0.20-0.30 | MM2,MAXSAT |
| 0.30-0.35 | MAXSAT,NK1 |
| 0.35-0.40 | MM{3,7,5,14,18},MAXSAT,NK2,NK3 |
| 0.40-0.45 | NK4,NIAH,MAXSAT |
| 0.45-0.50 | MM17,MM13,NK5-9,RAND |
| 0.50-0.60 | MM15,RAND |
| 0.60-0.70 | $TRAP_4$ |
| 0.70-0.80 | $TRAP_3$ |
| 0.80-1.00 | $TRAP_2$,$TRAP_1$ |

a variable level of difficulty: the NK landscapes (Altenberg, 1997b) with $k$ =1–10, multimodal landscapes (MM1, MM2, etc.) with a varying number of local maxima (1–20), and trap functions ($TRAP_i$ where $i \in \{1, \cdots, l\}$ indicates the level of difficulty, $TRAP_1$ being the most difficult). Finally, to test our measure of difficulty on landscapes which were not induced by artificial problems, we also considered 12 random MAXSAT problems. For each problem, we consider only one global optimum. If a problem had more than one global optimum, we chose one at random to be the target solution.

Figure 8 shows the measured performance for our test problems vs. their predicted performance $h(V)$, i.e., their distance from $V_{\text{easy}}$. For easier visualisation, landscapes were ordered on the basis of their predicted difficulty (from easiest to hardest). The correlation coefficient between observed and predicted difficulty is 0.82, which is very good.

Table 1 gives our (predicted) measure of difficulty $h(V)$ by problem, showing that our measure fully matches results obtained in previous research. Interestingly, the table shows that multimodality, as reported by many others before, is not a good indicator of problem difficulty. For example, a landscape with 3 local maxima has the same expected difficulty as a landscape with 7 local maxima, while a landscape with 15 local maxima is more difficult than a landscape with 18.

The table also confirms that the NK model is not appropriate for the study of problem difficulty because problems with a $k > 2$ are already very difficult (Jansen, 2001). Indeed, our measure suggests that the difficulty of such landscapes is close to random.

Different instances of the same problem might have different degrees of difficulty in the black-box scenario (Jansen, 2001). Indeed, the predicted difficulty for different instances of the MAXSAT problems varies from 0.2 (easy) to 0.45 (difficult).

### 7.2.2 Hardness of known counterexamples for other difficulty measures

In the previous section we showed that, for a broad family of problems, our method predicts accurately problem difficulty. In this section we test our framework on three problems where other measures of difficulty have been shown to fail.

Naudts and Kallel (Naudts and Kallel, 2000) constructed a simple problem consisting of a deceptive mixture of Onemax and Zeromax, where both the fitness distance correlation (FDC) measure and the sitewise optimisation measure (a generalisation for the FDC and epistasis suggested in the same paper) failed to correctly predict performance. For this class of problems, the higher the mixture coefficient $m$, the harder the problem, yet no hardness measures was able to predict this. We performed experiments with this problem[14], with the control parameter $m$ varying from 1 (easy) to 9 (hard). The correlation between the predicted difficulty and the actual performance was 0.75. So, we were largely able to capture the difference in performance as the parameter $m$ varied.

Jansen (Jansen, 2001) showed that the fitness distance correlation of the ridge function is very small. Yet, this is an easy problem for a hill climber. So, we decided to apply our method to this function as well. The distance of the ridge function to the optimal landscape is 0.84, which indicates a very difficult problem. Indeed, the GA was not able to find the solution in 100 generations. A problem that is easy for a hill climber is not necessarily easy for a recombinative GA. This is where other measures of problem difficulty tend to fail. However, this is not a problem for our framework since our measure of difficulty is specific to the algorithm being used.

Jansen (Jansen, 2001) gave two counter examples to the bit-wise epistasis measure of difficulty. The first one was the NIAH which we already discussed extensively before. The second was the leading-one function. The distance of the leading one function from the optimum landscape is 0.36, which predicts well the performance shown by a GA.

### 7.2.3 Improving the prediction

In the previous section the framework was tested explicitly on counterexamples for other measures of difficulty, and was shown to provide reliable estimates. In this section we enhance the accuracy of our measure.

In a GA with uniform crossover, the values of the entries in the performance landscape tend to grow as the relative distance of solution from the global optimum decreases. Based on this information, which is obtained from small problems, we modified our difficulty measure and, instead of using the standard $d(V_1, V_2)$, we used the following weighted distance measure

$$d'(V_1, V_2) = \frac{1}{n} \sum_i w_i |v_{1_i} - v_{2_i}|. \tag{26}$$

where the $w_i$'s are weights associated to the entries of the information landscape. The weights were set to be equal to the relative distance of the two solutions from the optimum. Then we reassessed problem difficulty using this weighted distance between landscapes.

The correlation for the first set of problems (Section 7.2.1) was improved from 0.82 to 0.86, while that of the deceptive mixture was improved from 0.75 to 0.78.

## 7.3 Performance symmetry under problem negation

This section gives empirical evidence supporting a general property which emerges from the information landscape framework: the symmetry between the performance of an algorithm on a landscape and the performance of the algorithm on the negation of the landscape (Section 5.4).

We started again considering information landscapes over the search space of all binary strings of length 3, using the same experimental setup as in Section 7.1. Figure 9 shows the performances for 100 random landscapes, sorted from worst to best, against the performance of the negation of

[14]The GA used in (Naudts and Kallel, 2000) is different from the one used here.

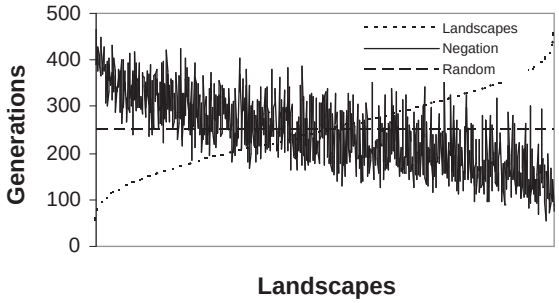



Figure 9: Symmetry between the measured performance on a landscape and the measured performance on the negation of the landscape (100 random landscapes).

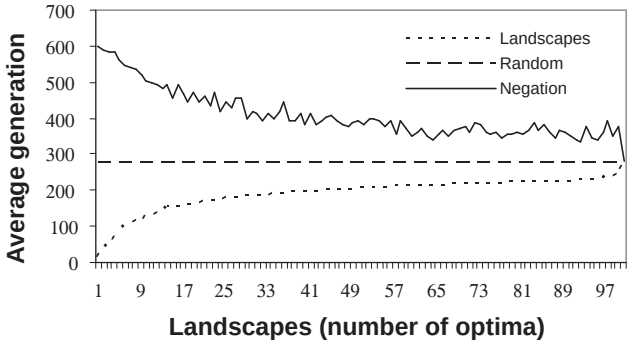



Figure 10: Empirical plot of the performance of a GA over 100 selected problems (dotted line). The solid line gives the performance on the negated landscape corresponding to each point in the dotted line.

those landscapes. We can clearly see the symmetry predicted by Equation 19 with respect to the performance over a NIAH landscape ($c_0$). The correlation between the performance measured for a landscape and the performance measured for the negation of the landscape is -0.775.

In order to demonstrate this effect in more realistic scenarios we used a slightly different version of GA. The size of the strings in this experiment was 16 bits, and, so, the search space included 65,536 points. The mutation rate was 0.15, the population size was 200 and the maximum number of generations was 600. This time as a performance measure we used the average number of generations required to find the global optimum.

We measured the performance of a GA over problems with an increasing level of multi-modality, which is a rough indicator of difficulty even though it is not very accurate. We used problems with 0 to 100 local optima. For each level of difficulty, Figure 10 reports the average of 500 different problems against the average of the negation of those problems. Equation 19 predicts a symmetry between a landscape and its negation with respect to the performance over a random landscape. The performance over a random landscape was measured as the average of 20 random landscapes of the same size. The correlation coefficient (-0.96) clearly shows that the symmetry exists also for larger, more realistic search spaces.

## 7.4 Inferring properties of specific search algorithms

In Section 6 we argued that our framework can be used to predict problems difficulty and in the previous sections we provided experimental evidence that, although limited to the case of a simple GA, strongly corroborated our claims. In this section we explore other ways in which the information landscape framework can aid the study of search algorithms.

We exemplify this using two versions of a simple GA: one based on one-point crossover, the other based on uniform crossover. Crossover was used with 100% probability. Again, the takeover time was used as the performance measure. We used a population size of 12. The maximum number of generations was 400. The search on each landscape was repeated 100 times. The results reported below are averages of those runs. The target solution, the string 111, was excluded from the first generation. Performance was computed for a sample of 100 valid landscapes of degree 1 (full information). In order to estimate the performance landscape (Equation 6) we did regression on the results obtained from running the GA over such landscapes.

The results of our experiments are reported in the remainder of this section. Section 7.4.1 shows that the performance landscape can be used to create case studies, and, in particular, to generate the easiest and most difficult problems for a simple GA. In the same section, we validate the predictions of the framework using static schema analysis. Section 7.4.2 describes how we can use the performance landscape directly to infer properties of the search operators.

We would like to emphasise that the methods introduced in this section do not require any explicit knowledge about the algorithms being analysed. As explained previously, computing the performance landscape is a, computationally heavy, but, conceptually, simple, 100% empirical procedure.

### 7.4.1 Automatic creation of case studies

As we mentioned earlier, very easy landscapes or very difficult ones have frequently been used in the literature to study search algorithms. Easy landscapes provide an intuition as to in which scenarios the algorithm performs well. Usually, it is easier to construct a theory restricted to those scenarios and validate it with empirical results. Examples of this are the attempts to construct easy landscapes such as, for example, the royal road function (Forrest and Mitchell, 1992) and the extensive investigation of problems like Onemax (Droste, 2004). Difficult landscapes provide similar intuitions for situations where an algorithm fails.

In Section 4.1 we used the notion of performance landscape to estimate the easiest possible landscape (Equation 7) and the hardest possible landscape (Equation 8) for an algorithm. To corroborate these theoretical predictions we computed the performance landscape for our GAs and constructed the corresponding $V_{\max}$ and $V_{\min}$ landscapes. When possible we then used Equation 2 to build the corresponding rank-based fitness functions, and did a static schema analysis. Schema analysis is known to be a good tool to analyse the search conducted by the crossover operator. Static schema analysis predicts that an easy problem is one that is characterised by the fact that any schema that contains the optimum (in our case, the string 111) will have higher fitness than its "competitors". The competitors of a schema are schemata with the same "don't care" symbols but with different defining symbols (i.e., schemata in the same partition) as the schema of interest.

Table 2 gives the rank-based fitness function reconstructed from the best information landscape for one-point crossover. Table 3 gives a static schema analysis for this fitness function. The table reveals that the schemata 11*, 1*1, *11, **1, *1*, and 1** which contain the global optimum 111 have a higher static fitness (which is simply the average of the fitness of all the strings in the search space matching the schema) than that of the other schemata in their partitions. Thus, the predictions made using the performance landscape are fully consistent with those of traditional schema analysis. In addition, these predictions were made without using any knowledge about the algorithm beyond its performance landscape.

The optimum landscape which was predicted for uniform crossover was also extremely easy to solve for the GA, but, formally, it is an invalid landscape (one that cannot be derived from a fitness function – see Section 3). Therefore, we are not able to present a schema analysis for it.

So far we considered landscapes with a degree of information equal to 1. However, the performance landscape can also be used to infer optimum landscapes with a degree of information smaller than 1. All we need to do to achieve an optimum landscape of degree $d$ is set to 0.5 a fraction $(1 - d)$ of the entries in the landscape $V_{\max}$ computed using Equation 7. The entries to modify are those which give the least positive contribution to performance, i.e., those for which $c_i$ is minimum.

Table 2: Rank-based fitness of the optimal information landscape predicted using the performance landscape for a simple GA with one-point crossover

| String | $f_{\text{rank}}$ |
|---|---|
| 000 | 0 |
| 010 | 1 |
| 100 | 2 |
| 001 | 3 |
| 101 | 4 |
| 011 | 5 |
| 110 | 6 |
| 111 | 7 |

Table 3: Static schema analysis of the optimal rank-based fitness function (Table 2) predicted using the performance landscape for a simple GA with one-point crossover. The strings ###, ##*, etc. represent schema partitions (the symbol # stands for a defining symbol, normally 0 or 1). Columns list the schemata competing in each partition along with their static fitness. Schemata in boldface are those with the highest fitness in each partition.

| Schema Partitions | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ### | $\bar{f}_{\text{rank}}$ | ##* | $\bar{f}_{\text{rank}}$ | #*# | $\bar{f}_{\text{rank}}$ | *## | $\bar{f}_{\text{rank}}$ | **# | $\bar{f}_{\text{rank}}$ | *#* | $\bar{f}_{\text{rank}}$ | #** | $\bar{f}_{\text{rank}}$ | *** | $\bar{f}_{\text{rank}}$ |
| **111** | 7 | **11*** | 6.5 | **1*1** | 5.5 | ***11** | 6 | ****1** | 4.75 | ***1*** | 4.75 | ***1**** | 4.75 | ****** | 3.5 |
| | | 01* | 3 | 0*1 | 4 | *01 | 3.5 | **0 | 2.25 | *0* | 2.25 | 0** | 2.25 | | |
| | | 10* | 3 | 1*0 | 4 | *10 | 3.5 | | | | | | | | |
| | | 00* | 1.5 | 0*0 | 0.5 | *00 | 1 | | | | | | | | |

We used this procedure to investigate whether the optimal landscapes predicted by our model would coincide with other known GA-easy landscapes, such as the Onemax problem, which is probably the most studied problem of such a kind. We chose, therefore, to find the optimal landscape of degree 0.71 (the degree of Onemax). The same results were obtained for both the one-point crossover performance landscape and the uniform crossover one: *the predicted optimal landscape was indeed Onemax.*

Finally, using Equation 8, we constructed the worst landscapes for our algorithms. Table 4 gives the rank-based fitness function reconstructed from the worst information landscape for one-point crossover. Table 5 gives a static schema analysis for this fitness function. The table reveals that the static fitness of each schema containing the global optimum is smaller than the static fitness of one of its competitors. Thus, *the worst information landscape corresponds to a fully deceptive landscape.* Like before, this result was obtained entirely automatically, without using any explicit knowledge about the search algorithm's operations.

Table 4: Rank-based fitness function associated to the worst possible information landscape predicted using the performance landscape for a simple GA with one-point crossover.

| String | $f_{\text{rank}}$ |
|---|---|
| 000 | 6 |
| 010 | 5 |
| 100 | 4 |
| 001 | 3 |
| 101 | 2 |
| 011 | 1 |
| 110 | 0 |
| 111 | 7 |

Table 5: Static schema analysis of the fitness function in Table 4.

| Schema Partitions | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ### | $\bar{f}_{\text{rank}}$ | ##* | $\bar{f}_{\text{rank}}$ | #*# | $\bar{f}_{\text{rank}}$ | *## | $\bar{f}_{\text{rank}}$ | **# | $\bar{f}_{\text{rank}}$ | *#* | $\bar{f}_{\text{rank}}$ | #** | $\bar{f}_{\text{rank}}$ | *** | $\bar{f}_{\text{rank}}$ |
| **111** | 7 | 11* | 3.5 | 1*1 | 4.5 | *11 | 4 | **1** | 3.25 | *1* | 3.25 | 1** | 3.25 | ******* | 3.5 |
| | | 01* | 3 | 0*1 | 2 | *01 | 2.5 | ****0** | 3.75 | ***0*** | 3.75 | **0**** | 3.75 | | |
| | | 10* | 3 | 1*0 | 2 | *10 | 2.5 | | | | | | | | |
| | | **00*** | 4.5 | **0*0** | 5.5 | ***00** | 5 | | | | | | | | |

### 7.4.2 Analysis of the performance landscape

In the previous section we have shown that the optimal and worst information landscapes calculated using our technique provide useful information about the *general* behaviour of an algorithm. In this section we show additional properties of the GA that can be inferred using a direct analysis of the performance landscape.

The coefficients $c_i$ in the performance landscape are estimated using empirical results and, therefore, they can be noisy. In order to reduce the effect of noise in our analysis, we use a clustering algorithm on the $c_i$'s. By analysing the clusters, we can then make robust inferences on the important properties of an algorithm. The original $c_i$ values can then be used for finer-granularity analyses.

In order to demonstrate how this can be done, we applied the $k$-means algorithm to cluster the entries of the two performance landscapes used in Section 7.4.1. Tables 6(a) and (b) show the landscapes and the results of the clustering. The two tables present all possible pairs of binary strings of length 3 with the absolute values of their associated $c_i$ from the performance landscapes of one-point crossover and uniform crossover. The pairs of strings are grouped according to the optimal clusters provided by the $k$-means algorithm. The clusters are sorted by the absolute value of $c_i$.

Tables 6(a)–(b) can be used to infer interesting properties of the search operators. Each entry represents the significance of choosing a particular solution over another. Both tables clearly show that different comparisons between points in the search space have very different significance for a GA (e.g., the values in Table 6(a) vary over about 2.5 orders of magnitude from the tiny 0.06 to the huge 16.1).

Analysing these entries we can also infer the relative importance of making the "correct" decision for different strings. For example, if we look at the strings 101 and 000 in Table 6(a), we can see that while the former should win each comparison in order for performance to be maximised, the latter should lose all of them. This suggests that the search operators are more likely to generate the optimum when using the string 101 than the string 000, which is exactly what we would expect knowing how selection and crossover interact in a GA. However, this result is obtained, again, *without* knowing how the algorithm operates.

To perform a more systematic analysis of the relative importance of solutions, from the performance landscapes of one-point crossover and uniform crossover we calculated the relative importance of all strings in the search space using the sum-based and rank-based performance vectors introduced in Section 4.2. Tables 7 and 8 show the results of the calculations. These clearly show that there are natural importance clusters for the string in the search space, and that these are different for the two crossovers. Figure 11 gives the resulting clusters as a partial order. Although, qualitatively, these clusters make sense, an important question is: how can we empirically validate them?

Empirically one can roughly estimate the importance, for the purpose of increasing performance, of choosing string $x$ over some other string by computing the average probability that when string $x$ is one of the parents, the crossover operator will generate the optimum. That is we can use the function

$$I(x) = \Pr\{x \to 111\} \approx \frac{1}{|X|} \sum_i \Pr\{x, x_i \to 111\}, \tag{27}$$

Table 6: Performance landscapes of GAs based on uniform (a) and one-point (b) crossover and the clusters obtained with the $k$-means algorithm. All performance landscape values are shown as absolute values. The strings that should win the comparisons in order for the algorithm to choose well in relation to the objective of finding the global optimum are underlined (except when $|c_i| < 1$ since in this case $c_i$ is too noisy to trust its sign).

(a) Uniform Crossover

| $\|c_i\|$ | Entry |
|---|---|
| 0.06 | 010, 100 |
| 0.21 | 011, 101 |
| 0.46 | 001, 100 |
| 0.77 | 011, 110 |
| 0.82 | 101, 110 |
| 0.84 | 001, 010 |
| 5.23 | 000, <u>100</u> |
| 5.49 | 000, <u>001</u> |
| 5.53 | 000, <u>010</u> |
| 10.1 | <u>011</u>, 100 |
| 10.5 | 001, <u>110</u> |
| 10.7 | 010, <u>101</u> |
| 12.0 | 001, <u>011</u> |
| 12.3 | 100, <u>101</u> |
| 12.5 | 100, <u>110</u> |
| 12.7 | 010, <u>011</u> |
| 12.7 | 010, <u>110</u> |
| 12.7 | 001, <u>101</u> |
| 15.0 | 000, <u>110</u> |
| 15.6 | 000, <u>011</u> |
| 16.1 | 000, <u>101</u> |

(b) One point Crossover

| $\|c_i\|$ | Entry |
|---|---|
| 0.09 | 011, 110 |
| 0.24 | 001, 100 |
| 2.70 | <u>001</u>, 010 |
| 2.80 | 101, <u>110</u> |
| 2.90 | 010, <u>100</u> |
| 2.95 | <u>011</u>, 101 |
| 3.83 | 000, <u>010</u> |
| 6.79 | 000, <u>100</u> |
| 7.46 | 000, <u>001</u> |
| 9.26 | <u>011</u>, 100 |
| 9.38 | 001, <u>110</u> |
| 9.63 | 100, <u>101</u> |
| 9.86 | 001, <u>101</u> |
| 11.8 | 010, <u>101</u> |
| 12.0 | 100, <u>110</u> |
| 12.8 | 001, <u>011</u> |
| 13.1 | 000, <u>101</u> |
| 17.9 | 000, <u>011</u> |
| 18.1 | 000, <u>110</u> |
| 19.1 | 010, <u>011</u> |
| 19.4 | 010, <u>110</u> |

Table 7: Sum-based and rank-based performance vectors induced from the performance landscape of simple GA with uniform crossover. The solution 111 is not present since it is excluded from the performance landscape. All entries with $|c_i| < 1$ were treated as 0 to reduce the effects of noise. These contributed 0.5 to $c_{\text{rank}}$ for the strings involved in the corresponding comparison.

| String | $c_{\text{sum}}$ | $c_{\text{rank}}$ |
|---|---|---|
| 000 | -62.95 | 0 |
| 010 | -30.57 | 2 |
| 100 | -29.87 | 2 |
| 001 | -29.71 | 2 |
| 101 | 52.4 | 5 |
| 011 | 50.4 | 5 |
| 110 | 50.7 | 5 |

Table 8: Sum-based and rank-based performance vectors induced from the performance landscape of simple GA with one-point crossover.

| String | $c_{\text{sum}}$ | $c_{\text{rank}}$ |
|---|---|---|
| 000 | -67.18 | 0 |
| 010 | -52.07 | 1 |
| 100 | -21.2 | 2.5 |
| 001 | -21.88 | 2.5 |
| 101 | 38.64 | 4 |
| 011 | 62.01 | 5.5 |
| 110 | 61.68 | 5.5 |

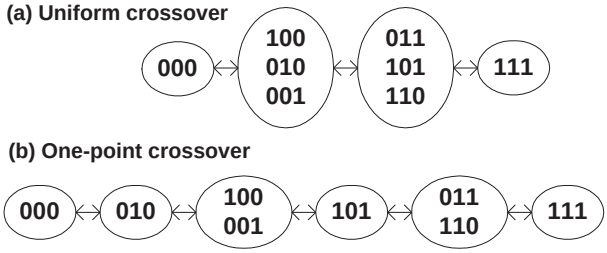



Figure 11: Partial order based on the relative importance of making correct decisions for different strings in the search space for uniform crossover (a) and one-point crossover (b).

where $\Pr\{x, x_i \rightarrow 111\}$ is the probability of generating the optimum string 111 when crossing over $x$ with the $i$th element of the search space, $x_i$. Naturally, this cannot account for the full dynamics of the algorithm: the approximation assumes that all strings are equally represented in the population and it models what happens over just one generation. However, the estimate may be sufficiently accurate to give us an empirical indication of the relative importance of each point in the search space and to corroborate the results in Tables 7 and 8 and Figure 11.

Going back to Figure 11(a), which gives the results for uniform crossover, the three different importance groups can be easily explained using the estimate of $I(x)$ in Equation 27. Indeed, the probability of generating the optimum in one generation using any string from the group {011, 101, 110} is 0.109375. For the strings in the group {001,010,100} the probability is instead 0.046875. Finally, for the string 000 the probability is 0.015625. These results confirm the accuracy of the clusters produced with our framework.

As we stated earlier, Equation 27 does not account for the complexities of the algorithm. However, the performance landscape does not suffer from the same limitations. For example, it represents the behaviour of an algorithm over complete runs, not just one generation. Also, it makes no assumptions on the behaviour of the algorithm, e.g., it takes into account that real GAs do not sample all solutions in the search space with equal probability (an unrealistic assumption we had to formulate to get a rough estimate of $I(x)$ in Equation 27).

So, if the information landscape is estimated accurately using a large number of runs and an unbiased training set (which is the case in our experiments), a careful examination of its elements may reveal even very subtle properties of an algorithm. For example, let us focus on the third cluster in Table 6(a) which groups the entries corresponding to comparisons between strings in the group {011, 101, 110} with strings in the group {001,010,100}. The performance landscape elements $c_{001,110}$, $c_{010,101}$, and $c_{100,011}$ have slightly lower values (10.1–10.7) than the other entries in the cluster (which have values 12.0–12.7). This difference cannot easily be explained looking at one-generation behaviour, e.g., via Equation 27. However, it is an important difference. The explanation for it is that the strings 001 and 110 are complementary for the purpose of reaching the global optimum, 111. That is, the algorithm needs instances of both to create the global optimum. The same is true for the pairs 010/101 and 100/011. So, when the algorithm is forced

to choose between the two strings of a complementary pair, it picks the better one to continue the search, but it is forced to discards the important genetic material contained in the other string. So, performance is worse, than when the decision does not imply loss of important genetic material.

Figure 11(b) gives the importance groups for one-point crossover. Unlike uniform crossover, this operator is biased with respect to bit position. In particular, recombining the strings 010 and 101 cannot produce the optimum. This difference can be explained using the estimate provided in Equation 27. The probability of creating the optimum for each string in the group {110, 011} is 0.1875, it is 0.125 for the string 101, it drops to 0.0625 for the group {100,001} and it is zero for the strings 010 and 000.[15] A finer-grain analysis of Table 6(b) reveals some additional interesting properties in this case as well. Firstly, as expected, the slightly negative effect of having to choose between complementary strings exists here too. Moreover, notice that the value of $c_{010,110}$, for example, is higher than $c_{000,110}$. That is, a landscape for which the string 110 wins over 000 but loses to 010 is more difficult than the other way around (110 loses to 000 but wins over 010). This might appear counter-intuitive. However, the string 010 is a possible future competitor of the string 001 (the complement of 110). Therefore, when considering more than one generation, choosing 110 while getting rid of a competitor for its complementary string increases the probability to produce the optimum over multiple generations.

Although the analyses we did in this section are based on the full knowledge of the performance landscape, and, therefore, can be performed only for small landscapes, the lessons we learned from such landscapes are numerous and general. Also, the previous analysis indicated how the performance vectors $c_{\text{rank}}$ and $c_{\text{sum}}$, which are much more compact representations for an algorithm, may contain very meaningful information. We believe that this may, in fact, be sufficient for many practical purposes. For example, a simple comparison between corresponding entries in the fitness function $f_{\text{rank}}$ in Table 2 and the rank-based performance vectors $c_{\text{rank}}$ in Tables 7 and 8 reveals the good correlation between the goodness of solutions (fitness) as seen from the point of view of the problem and the importance of solutions as seen from the point of view of an algorithm. Likewise, we see the anti-correlation between $c_{\text{rank}}$ and the fitness function in Table 4. These differences can easily be quantised via the simple scalar $f_{\text{rank}} \cdot c_{\text{rank}}$ (where obviously we omit the last entry of the fitness function, $f_{\text{rank}}(111)$).

# 8 Conclusions

In this paper we introduced a novel framework which has implications on both the general study of problems in the black-box scenario and the study of particular search algorithms. The framework is based on the notions of information landscape and performance landscape.

The information landscape (Section 3) captures the information available to an algorithm to conduct a search. This is quantified by the degree of information (Equation 4). A NIAH landscape has degree of information 0, meaning that it does not contain any information, and, so, any non-resampling search algorithm will perform equally well (or badly) on it. More generally, the less information a landscape contains, the more it resembles a NIAH, and, therefore, the closer the performance of any algorithm will be to that of random search. The performance landscape (Section 4), instead, represents the sensitivity of the performance of an algorithm to the decisions the algorithm has to make when choosing between pairs of solutions. The use of these two landscapes allows one to understand the interaction between algorithms and problems in very simple terms. For example, it allows us to determine automatically on which problems an algorithm will perform best and worst.

In Section 5 we discussed the contributions of this framework to the understanding of search in the black box scenario. A geometric interpretation based on the notion of scalar product was proposed which allows us to visualise and understand in simple terms the connection between search algorithms and problems (Section 5.1). In Section 5.2 we explicitly linked our framework

[15] Equation 27 predicts that the probability to generate the optimum using the strings 000 and 010 is identical and it is 0, even if string 010 is clearly closer to the optimum. This is because the equation estimates only what happens over one generation.

to the NFLTs, showing that it is consistent with these. Furthermore, our framework allowed us to see explicitly the differences between three different notions of randomness in search: search on a random landscape, stochastic search on a NIAH landscape, and true random search (Section 5.3). Finally, Section 5.4 illustrated the connection between easy and difficult problems showing that, to a first order approximation, one class is just the negation of the other (Equation 18). The NFLTs prove that for every problem on which an algorithm performs well there exists a problem on which it performs badly, but they do not tell us which. The information landscape framework goes one step further: it gives us a precise definition of that problem (of course, within the accuracy of our first order approximation to the performance function).

The information landscape framework can be applied to the study of particular search algorithms as well as the general study of search in the black box scenario. Section 6 shows how the framework can be used to derive a predictive measure to problem difficulty. Our difficulty measure $h(V)$ (in Equations 20 and 21 and later improved in Equation 26) uses the distance of a landscape from the optimal landscape to give an indication for the alignment or match between a problem and a search algorithm. This makes sense because the optimal landscape (Equation 7) is maximally aligned with the search algorithm by definition. So, the further a landscape is from the optimal landscape the worse the performance of the algorithm will be. The distance of both random and NIAH information landscapes from the optimal landscape is 0.5, indicating that they are both hard. There are, however, landscapes, such as deceptive landscapes, that can be even harder since they are completely anti-correlated with the regularities and expectations of the algorithm.

Section 7 extensively validated all the aforementioned aspects of the theory. The validation was done considering a large variety of problems including: all possible 3-bit problems in Section 7.1; Onemax, NIAH, random problems, NK landscapes, trap functions, unimodal and highly-multimodal problems, and MAXSAT problems in Section 7.2.1; several functions which are know to be problematic for other frameworks in Section 7.2.2; and negations of problems (Section 7.3). Also, in Section 7.4 we showed how the performance landscape can be used to study both coarse- and fine-grain behaviours of specific algorithms and that the results are fully compatible with the predictions made using other theoretical methods, including the static schema analysis (Section 7.4.1) and probabilistic models of recombination (Section 7.4.2).

The results of all our experiments with the framework have been remarkably encouraging. The information landscape framework is general and it is defined irrespective of the specific search algorithm under consideration. However, in our empirical validation we limited ourselves to using only simple GAs. So, one may wonder: to which extent is the method general? In future research we intend to extend the empirical validation to other algorithms. However, we can already get a feel for the scope of the method by considering the underlying assumptions of our model.

Our first assumption is that all the information that the algorithm uses is captured in the information landscape. Clearly the information landscapes throws away some of the information available in the fitness landscape. In particular, it considers only relative fitness values. For many algorithms this is all is required. For example, in GAs, where fitness/problem information is used only in the selection phase, the information landscape contains all the data necessary to run an algorithm based on binary tournament selection. Also, when the information landscape is induces by a fitness function, the information matrix defines implicitly the rank of each solution and, so, it includes all the data necessary to perform tournaments of bigger sizes as well as to perform rank selection.[16] Also, beyond the GA world, many other algorithms, such as $(1+1)$ evolutionary strategies, stochastic hill climbing, binary particle swarm optimisation, differential evolution, etc., make only use of relative fitness values. Therefore, the information landscape really captures the features of a fitness landscape that are relevant from an algorithm's perspective, for a huge variety of practical algorithms. It is also possible that information landscapes provide reasonable approximations of the information available to search algorithms that use absolute fitness values, such as a GA with fitness-proportionate selection. This is another issue we want to explore in future research.

[16]The information matrix is all one would need to apply a sorting algorithms, like quicksort, to order any subset of elements of the search space (e.g., any tournament set and any population).

The second assumption of the model is that a first order, linear approximation of the performance function $P(V)$ is sufficiently accurate for most practical purposes. We have thoroughly tested this for the class of GAs, with very good results. Naturally, we cannot be sure of the approximation level for other algorithms, but we expect the approximation to be reasonable for most population-based algorithms, since for these algorithms many elements of the performance landscape are non-zero.

In future research we intend to test the generality of these assumptions for a variety of different metaheuristics.

## Acknowledgements

The authors would like to thank Chris Stephens, Bill Langdon, Thomas Jensen and Marc Toussaint for useful discussions and comments.

## References

Altenberg, L. (1997a). Fitness distance correlation analysis: An instructive counterexample. In Bäck, T., editor, *Proceedings of the 7th International Conference on Genetic Algorithms, East Lansing, MI, USA, July 19-23, 1997*, pages 57–64. Morgan Kaufmann.

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

Asoh, H. and Mühlenbein, H. (1994). On the mean convergence time of evolutionary algorithms without selection and mutation. In Davidor, Y., Schwefel, H.-P., and Männer, R., editors, *Parallel Problem Solving from Nature - PPSN III, International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature, Jerusalem, Israel, October 9-14, 1994, Proceedings*, volume 866 of *Lecture Notes in Computer Science*, pages 88–97. Springer.

Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308.

Borenstein, Y. and Poli, R. (2004). Fitness distributions and GA hardness. In Yao, X., Burke, E. K., Lozano, J. A., Smith, J., Guervós, J. J. M., Bullinaria, J. A., Rowe, J. E., Tiño, P., Kabán, A., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference, Birmingham, UK, September 18-22, 2004, Proceedings*, volume 3242 of *Lecture Notes in Computer Science*, pages 11–20. Springer.

Culberson, J. C. (1998). On the futility of blind search: An algorithmic view of "no free lunch". *Evolutionary Computation*, 6(2):109–127.

Davidor, Y. (1990). Epistasis variance: A viewpoint on GA-hardness. In Rawlins, G. J. E., editor, *Proceedings of the First Workshop on Foundations of Genetic Algorithms. Bloomington Campus, Indiana, USA, July 15-18 1990.*, pages 23–35. Morgan Kaufmann.

Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization*. MIT Press (Bradford Books).

Droste, S. (2004). Analysis of the (1+1) EA for a noisy onemax. In Deb, K., Poli, R., Banzhaf, W., Beyer, H.-G., Burke, E. K., Darwen, P. J., Dasgupta, D., Floreano, D., Foster, J. A., Harman, M., Holland, O., Lanzi, P. L., Spector, L., Tettamanzi, A., Thierens, D., and Tyrrell, A. M., editors, *Genetic and Evolutionary Computation - GECCO 2004, Genetic and Evolutionary Computation Conference, Seattle, WA, USA, June 26-30, 2004, Proceedings, Part I*, volume 3102 of *Lecture Notes in Computer Science*, pages 1088–1099. Springer.

Droste, S., Jansen, T., and Wegener, I. (2003). Upper and lower bounds for randomized search heuristics in black-box optimization. *Electronic Colloquium on Computational Complexity (ECCC)*, (048).

English, T. M. (2000a). Optimization is easy and learning is hard in the typical function. In Zalzala, A., Fonseca, C., Kim, J.-H., and Smith, A., editors, *Proceedings of the 2000 Congress on Evolutionary Computation (CEC 2000)*, pages 924–931. IEEE Press.

English, T. M. (2000b). Practical implications of new results in conservation of optimizer performance. In Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Guervós, J. J. M., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VI, 6th International Conference, Paris, France, September 18-20, 2000, Proceedings*, volume 1917 of *Lecture Notes in Computer Science*, pages 69–78. Springer.

English, T. M. (2004). On the structure of sequential search: Beyond "no free lunch". In Gottlieb, J. and Raidl, G. R., editors, *Evolutionary Computation in Combinatorial Optimization, 4th European Conference, EvoCOP 2004, Coimbra, Portugal, April 5-7, 2004, Proceedings*, volume 3004 of *Lecture Notes in Computer Science*, pages 95–103. Springer.

Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966). *Artificial Intelligence through Simulated Evolution*. Wiley, New York.

Forrest, S. and Mitchell, M. (1992). Relative building-block fitness and the building block hypothesis. In Whitley, L. D., editor, *Proceedings of the Second Workshop on Foundations of Genetic Algorithms. Vail, Colorado, USA, July 26-29 1992.*, pages 109–126. Morgan Kaufmann.

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

Goldberg, D. E. (1993). Making genetic algorithm fly: a lesson from the Wright brothers. *Advanced Technology For Developers*, 2:1–8.

Grefenstette, J. J. (1992). Deception considered harmful. In Whitley, L. D., editor, *Proceedings of the Second Workshop on Foundations of Genetic Algorithms. Vail, Colorado, USA, July 26-29 1992.*, pages 75–91. Morgan Kaufmann.

Guo, H. and Hsu, W. H. (2003). GA-hardness revisited. In Cantú-Paz, E., Foster, J. A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Standish, R. K., Kendall, G., Wilson, S. W., Harman, M., Wegener, J., Dasgupta, D., Potter, M. A., Schultz, A. C., Dowsland, K. A., Jonoska, N., and Miller, J. F., editors, *Genetic and Evolutionary Computation - GECCO 2003, Genetic and Evolutionary Computation Conference, Chicago, IL, USA, July 12-16, 2003. Proceedings, Part II*, volume 2724 of *Lecture Notes in Computer Science*, pages 1584–1585. Springer.

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.

H"ohn, C. and Reeves, C. R. (1996). The crossover landscape for the onemax problem. In Alander, J. T., editor, *Proc. of the Second Nordic Workshop on Genetic Algorithms and their Applications (2NWGA)*, pages 27–43, Vaasa, Finnland. Department of Information Technology and Production Economics, University of Vaasa.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, USA.

Igel, C. and Toussaint, M. (2004). A no-free-lunch theorem for non-uniform distributions of target functions. *Journal of Mathematical Modelling and Algorithms*, 3(4):313–322.

Jansen, T. (2001). On classifications of fitness functions. In *Theoretical aspects of evolutionary computing*, pages 371–385. Springer-Verlag, London, UK.

Jansen, T. and Wegener, I. (2000). On the choice of the mutation probability for the (1+1) EA. In Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Guervós, J. J. M., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VI, 6th International Conference, Paris, France, September 18-20, 2000, Proceedings*, volume 1917 of *Lecture Notes in Computer Science*, pages 89–98. Springer.

Jones, T. (1995a). *Evolutionary Algorithms, Fitness Landscapes and Search*. PhD thesis, The University of New Mexico, Albuquerque, NM.

Jones, T. (1995b). One operator, one landscape. URL:citeseer.ist.psu.edu/jones95one.html.

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

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

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

Langdon, W. B. and Poli, R. (2002). *Foundations of Genetic Programming*. Springer-Verlag.

Lee, C.-Y. and Antonsson, E. K. (2000). Variable length genomes for evolutionary algorithms. In Whitley, L. D., Goldberg, D. E., Cantú-Paz, E., Spector, L., Parmee, I. C., and Beyer, H.-G., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '00), Las Vegas, Nevada, USA, July 8-12, 2000*, page 806. Morgan Kaufmann.

Naudts, B. (1998). *Measuring GA-hardness*. PhD thesis, University of Antwerpen, Antwerpen, Netherlands.

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

Nix, A. E. and Vose, M. D. (1992). Modeling genetic algorithms with Markov chains. *Annals of Mathematics and Artificial Intelligence*, 5(1):79–88.

P.F.Stadler and C.R.Stephens (2002). Landscapes and effective fitness. *Comments Theori. Biol.*, 8:389–431.

Poli, R., Stephens, C. R., Wright, A. H., and Rowe, J. E. (2002). On the search biases of homologuous crossover in linear genetic programming and variable-length genetic algorithms. In Langdon, W. B., Cantú-Paz, E., Mathias, K., Roy, R., Davis, D., Poli, R., Balakrishnan, K., Honavar, V., Rudolph, G., Wegener, J., Bull, L., Potter, M. A., Schultz, A. C., Miller, J. F., Burke, E., and Jonoska, N., editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 868–876, New York. Morgan Kaufmann Publishers.

Prügel-Bennett, A. and Shapiro, J. L. (1994). An analysis of genetic algorithms using statistical mechanics. *Physical Review Letters*, 72:1305–1309.

Rana, S. (1998). *Examining the Role of Local Optima and Schema Processing in Genetic Search*. PhD thesis, Colorado State University, Colorado, U.S.A.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann–Holzboog, Stuttgart.

Reeves, C. R. (1999). Landscapes, operators and heuristic search. *Annals of Operations Research*, 86:473 – 490.

Reidys, C. M. and Stadler, P. F. (2002). Combinatorial landscapes. *SIAM Rev.*, 44(1):3–54.

Rothlauf, F. (2002). *Representations for Genetic and Evolutionary Algorithms*. Heidelberg: Springer.

Schumacher, C., Vose, M. D., and Whitley, L. D. (2001). The no free lunch and problem description length. In Spector, L., Goodman, E. D., Wu, A., Langdon, W., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. H., and Burke, E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 565–570, San Francisco, California, USA. Morgan Kaufmann.

Schwefel, H.-P. (1981). *Numerical Optimization of Computer Models*. Wiley, Chichester.

Simões, A. and Costa, E. (2002). Parametric study to enhance the genetic algorithm's performance when using transformation. In Langdon, W. B., Cantú-Paz, E., Mathias, K. E., Roy, R., Davis, D., Poli, R., Balakrishnan, K., Honavar, V., Rudolph, G., Wegener, J., Bull, L., Potter, M. A., Schultz, A. C., Miller, J. F., Burke, E. K., and Jonoska, N., editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, New York, USA, 9-13 July 2002*, page 697. Morgan Kaufmann.

Stephens, C. R. and Waelbroeck, H. (1999). Schemata evolution and building blocks. *Evolutionary Computation*, 7(2):109–124.

Vose, M. D. (1998). *The Simple Genetic Algorithm: Foundations and Theory*. MIT Press, Cambridge, MA, USA.

Wegener, I. (2003). Towards a theory of randomized search heuristics. In Rovan, B. and Vojtás, P., editors, *MFCS*, volume 2747 of *Lecture Notes in Computer Science*, pages 125–141. Springer.

Wegener, I. (2004). Randomized search heuristics as an alternative to exact optimization. In Lenski, W., editor, *Logic versus Approximation, Essays Dedicated to Michael M. Richter on the Occasion of his 65th Birthday*, volume 3075 of *Lecture Notes in Computer Science*, pages 138–149. Springer.

Weinberger, E. D. (1991). Fourier and Taylor series on fitness landscapes. *Biol. Cybern.*, 65:321–330.

Wolpert, D. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Trans. Evolutionary Computation*, 1(1):67–82.

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