Problem Understanding through Landscape Theory

Francisco Chicano
University of Málaga, Spain
chicano@lcc.uma.es

Gabriel Luque
University of Málaga, Spain
gabriel@lcc.uma.es

Enrique Alba
University of Málaga, Spain
eat@lcc.uma.es

ABSTRACT

In order to understand the structure of a problem we need to measure some features of the problem. Some examples of measures suggested in the past are autocorrelation and fitness-distance correlation. Landscape theory, developed in the last years in the field of combinatorial optimization, provides mathematical expressions to efficiently compute statistics on optimization problems. In this paper we discuss how can we use landscape theory in the context of problem understanding and present two software tools that can be used to efficiently compute the mentioned measures.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Theory, Algorithms

Keywords

Fitness Landscapes, Elementary Landscapes, Problem Understanding, Quadratic Assignment Problem, Unconstrained Quadratic Optimization

1. INTRODUCTION

Landscape theory is a set of definitions and theorems that allows one to analyze optimization problems in connection with a neighborhood structure defined over the search space. We are interested in the applications of the theory to Combinatorial Optimization. However, this theory has applications in Chemistry [19], Biology [24] and Physics [12]. One of the main goals of the theory is to better understand the structure of the optimization problems. Thanks to this deeper understanding we are supposed to be able to define new search operators, search strategies or even determine the optimal values for the parameters of the algorithms used to solve a given problem.

GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.

There is a special kind of *landscapes*, called *elementary landscapes*, with nice properties. In particular, they are characterized by Grover's wave equation [14]:

$$\operatorname*{avg}_{y\in N(x)}\{f(y)\} = f(x) + \frac{k}{d}\left(\bar{f} - f(x)\right), \qquad (1)$$

where d is the neighborhood size, $|N(x)|$, which we assume the same for all the solutions in the search space, $\bar{f}$ is the average value of the objective function in the whole search space and k is a constant. Either k or $\bar{f}$ can usually be efficiently and accurately computed from the problem data or by a random sampling of the search space.

The advantage of an expression like (1) is that it allows one to compute a statistics (the average in the neighborhood) from the value of a function in x. We must highlight here that if (1) were not true, we would need to evaluate every solution in $N(x)$ in order to compute the average in the neighborhood. Thus, landscape theory gives us results that allow us to compute in an efficient way some non trivial statistics related to the distribution of the objective function. This property is present in most of the results of landscape theory, as we will see along the next sections.

In this paper we will focus mainly on three measures that can be efficiently computed using landscape theory. They are the autocorrelation, the fitness-distance correlation and the expected fitness after bit-flip mutation. The first two statistics have been considered as measures for problem hardness. None of them is a perfect measure of the hardness of a problem and both have been criticized in the past. However, they are used even in some recent works. In this paper we will see how they can be efficiently computed using landscape theory under some conditions and we provide software tools to do it. Regarding the expected fitness after bit-flip mutation, it has some links to runtime analysis, which is a direct measure of problem hardness. But, more interesting is the fact that the elementary decomposition of a combinatorial optimization problem has a one-to-one relationship with the expectation curves (depending on the probability of mutation) of bit-flip.

Some of the results provided by landscape theory can be implemented in a computer. Although the implementation of these algorithms requires a deep knowledge of the theory, they can be used with a minimum knowledge. That is, it is possible to enclose the knowledge provided by landscape theory in a code snippet. Some of these code snippets can be applied to any optimization problem but most of them are specific to a given problem. For example, the algorithms for efficiently and accurately computing the statistics mentioned

above can only be applied to concrete problems. During the last three years we have frequently implemented code snippets and algorithms to check empirically the results provided by the theory. We think the resulting software could be useful for researchers interested in problem understanding. For this reason, we present, as the main contribution of this paper, two of the implemented applications and provide links to them. In one case we provide a web application that everyone can use without installing anything and in the other case we provide a GUI-based application.

The organization of the paper is as follows. In Section 2 we provide some background on landscape theory. Sections 3, 4 and 5 discusses how landscape theory can be used to efficiently compute the autocorrelation, fitness-distance correlation and expected fitness after bit-flip mutation in a combinatorial optimization problem. Section 6 presents the software tools developed in the last years that implement the algorithms to efficiently compute the measures described and other parameters related to landscape theory. Finally, Section 7 concludes the work and describes future work.

2. BACKGROUND

In this section we present some fundamental results of landscape theory. We will only give a soft introduction to general concepts of landscape theory. We refer the reader interested in a deeper exposition of this topic to the survey in [17].

A *landscape* for a combinatorial optimization problem is a triple (X, N, f), where X is the solution set, $f : X \mapsto \mathbb{R}$ defines the objective function and N is the *neighborhood function*, which maps any solution $x \in X$ to the set $N(x)$ of points reachable from x. If $y \in N(x)$ then we say that y is a neighbor of x.

The pair (X, N) is called *configuration space* and can be represented using a graph $G(X, E)$ in which X is the set of vertices and a directed edge (x, y) exists in E if $y \in N(x)$ [5]. We can represent the neighborhood operator by its adjacency matrix

$$\boldsymbol{A}_{x,y} = \begin{cases} 1 & \text{if } y \in N(x), \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Any discrete function, f, defined over the set of candidate solutions can be characterized as a vector in $\mathbb{R}^{|X|}$. Any $|X| \times |X|$ matrix can be interpreted as a linear map that acts on vectors in $\mathbb{R}^{|X|}$. For example, the adjacency matrix $\boldsymbol{A}$ acts on function f as follows

$$\boldsymbol{A}\ f = \begin{pmatrix} \sum_{y \in N(x_1)} f(y) \\ \sum_{y \in N(x_2)} f(y) \\ \vdots \\ \sum_{y \in N(x_{|X|})} f(y) \end{pmatrix}. \tag{3}$$

The component x of this matrix-vector product can thus be written as:

$$(\boldsymbol{A}\ f)(x) = \sum_{y \in N(x)} f(y), \tag{4}$$

which is the sum of the function value of all the neighbors of x. When a neighborhood is regular the so-called *Laplacian matrix* is defined as $\boldsymbol{\Delta} = \boldsymbol{A} - d\boldsymbol{I}$. Stadler defines the class of *elementary landscapes* where the function f is an eigenvector (or eigenfunction) of the Laplacian up to an additive constant [18]. Formally, we have the following:

Definition 1. *Let (X, N, f) be a landscape and $\boldsymbol{\Delta}$ the Laplacian matrix of the configuration space. The landscape is said to be elementary if there exists a constant b, which we call* offset*, and an eigenvalue λ of $-\boldsymbol{\Delta}$ such that $(-\boldsymbol{\Delta})(f - b) = \lambda(f - b)$.*

We use eigenvalues of $-\boldsymbol{\Delta}$ instead of $\boldsymbol{\Delta}$ to have positive eigenvalues [5]. In connected neighborhoods (the ones we consider here) the offset b is the average value of the function over the whole search space: $b = \bar{f}$ [11]. In elementary landscapes, the average value $\bar{f}$ can be usually computed in a very efficient way using the problem data. That is, it is not required to do a complete enumeration over the search space.

Taking into account basic results of linear algebra, it is not difficult to prove that if f is elementary with eigenvalue λ, $af + b$ is also elementary with this same eigenvalue λ. Furthermore, in regular neighborhoods, if g is an eigenfunction of $-\boldsymbol{\Delta}$ with eigenvalue λ then g is also an eigenvalue of $\boldsymbol{A}$, the adjacency matrix, with eigenvalue $d - \lambda$. The average value of the fitness function in the neighborhood of a solution can be computed using the expression:

$$\underset{y \in N(x)}{\text{avg}}\{f(y)\} = \frac{1}{d}(\boldsymbol{A}\ f)(x). \tag{5}$$

If f is an elementary function with eigenvalue λ, then the average is computed as:

$$\begin{aligned} \underset{y \in N(x)}{\text{avg}}\{f(y)\} &= \underset{y \in N(x)}{\text{avg}}\{f(y) - \bar{f}\} + \bar{f} \\ &= \frac{1}{d}(\boldsymbol{A}\ (f - \bar{f}))(x) + \bar{f} = \frac{d - \lambda}{d}(f(x) - \bar{f}) + \bar{f} \\ &= f(x) + \frac{\lambda}{d}(\bar{f} - f(x)), \end{aligned} \tag{6}$$

and we get Grover's wave equation [14]. In the previous expression we used the fact that $f - \bar{f}$ is an eigenfunction of $\boldsymbol{A}$ with eigenvalue $d - \lambda$.

The wave equation makes it possible to compute the average value of the fitness function f evaluated over all of the neighbors of x using only the value $f(x)$, that is, it provides a way of computing non-trivial statistics with a low computational cost. The previous average can be interpreted as the expected value of the objective function when a random neighbor of x is selected using a uniform distribution. This is exactly the behaviour of the so-called *1-bit-flip* mutation [13]. It could seem that the restriction imposed by Grover's wave equation cannot be frequently found in optimization problems. However, there are some well-known NP-hard problems using common neighborhoods that are elementary landscapes. This is the case of the Not All Equals SAT problem, the Travelling Salesman Problem, the Graph Coloring problem, etc. The interested reader can find examples of elementary landscapes in [25, 26].

Certainly, a landscape (X, N, f) is not always elementary, but even in this case it is possible to characterize the function f as the sum of elementary landscapes, called *elementary components* of the landscape. When the neighborhood N of the landscape is symmetric there exists an orthogonal basis of the space of functions composed of elementary landscapes. Let us denote this basis with $\theta_{\lambda,i}$ where λ is the eigenvalue of the vector (function) and i is an index to distinguish the

different vectors with the same eigenvalue. Then a Fourier expansion of f is

$$f = \sum_{\lambda} \sum_{i} a_{\lambda,i} \theta_{\lambda,i},$$

where the values $a_{\lambda,i} = \langle \theta_{\lambda,i}, f \rangle$ are the Fourier coefficients. Using this Fourier expansion it is possible to compute the landscape decomposition by summing the terms with the same eigenvalue. Each elementary component can be computed as

$$f_{\lambda} = \sum_{i} a_{\lambda,i} \theta_{\lambda,i}. \tag{7}$$

A special case is that of f_0, the elementary landscape with $\lambda = 0$. If we assume that the neighborhood is connected then f_0 is the constant value $\bar{f}$. Finding the elementary components of a given optimization problem is not a trivial task. In the last years we can find some works devoted to the task of finding this decomposition for some well-known problems. Along the paper we will cite the corresponding works when needed. The reader interested in finding such a decomposition for her/his favourite optimization problem can find a methodology to do it in [11].

3. AUTOCORRELATION

The *autocorrelation coefficient* ξ of a problem is a parameter proposed by Angel and Zissimopoulos [1] that gives a measure of its ruggedness. The same authors showed later in an empirical study that ξ seems to be related with the performance of Simulated Annealing [2]. Another measure of ruggedness defined by García-Pelayo and Stadler is the *autocorrelation length* [12], denoted with ℓ. The autocorrelation length is specially important in optimization because of the *autocorrelation length conjecture*, which claims that in many landscapes the number of local optima M can be estimated by the expression [20]:

$$M \approx \frac{|X|}{|X(x_0, \ell)|}, \tag{8}$$

where $X(x_0, \ell)$ is the set of solutions reachable from x_0 in ℓ or less local movements. The previous expression is not exact, but an approximation. It can be useful to compare the estimated number of local optima in two instances of the same problem. In effect, for a given problem in which (8) is valid, the higher the value of ℓ the lower the number of local optima. In a landscape with a low number of local optima, a local search strategy can *a priori* find the global optimum using less steps.

In many problems, ξ and ℓ are directly related. That is, when one of them increases the other does the same. As an example we consider the Unconstrained Quadratic Optimization (UQO). We can observe in Figure 1 that both autocorrelation measures, ξ and ℓ, increase with W_2 (spectral coefficient, see below) and their values are between $n/4$ when $W_2 = 0$ and $n/2$ when $W_2 = 1$. However, the curve of ℓ is linear while the one of ξ is non-linear.

In the context of the autocorrelation length conjecture, this could explain why Angel and Zissimopoulos observed a better performance of SA when the problem instances had a higher value for ξ. Instances with higher ξ most probably would have higher ℓ and this most probably would mean a lower number of local optima in the instance, so it is easier

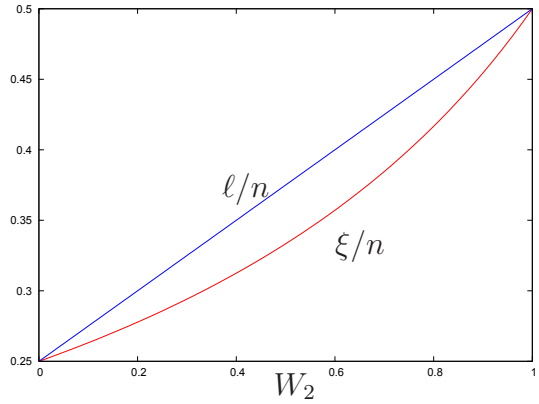

Figure 1: Value of ξ/n and ℓ/n against W_2.

for a local search algorithm like SA to solve the problem instance. A recent work based also in QAP supports this idea [9].

Let us discuss now how can we efficiently compute ξ and ℓ if we know the elementary landscape decomposition of a problem. Let us consider a random walk $\{x_0, x_1, \ldots\}$ on the solution space such that $x_{i+1} \in N(x_i)$. The Weinberger *autocorrelation function* r is defined as:

$$r(s) = \frac{\text{avg}\{f(x_t) f(x_{t+s})\}_{x_0,t} - \text{avg}\{f(x_t)\}^2_{x_0,t}}{\text{avg}\{f(x_t)^2\}_{x_0,t} - \text{avg}\{f(x_t)\}^2_{x_0,t}}, \tag{9}$$

where the averages are computed over all the starting solutions x_0 and all the solutions in the sequence [24]. Based on this function the autocorrelation coefficient and the autocorrelation length are defined as:

$$\xi = \frac{1}{1 - r(1)}, \tag{10}$$

$$\ell = \sum_{s=0}^{\infty} r(s). \tag{11}$$

Stadler [18] proved that if $f = \sum_i a_i \phi_i$ is a Fourier expansion of f in a landscape, then the autocorrelation function of f is given by

$$r(s) = \sum_{i \neq 0} \frac{a_i^2}{\sum_{j \neq 0} a_j^2} \left(1 - \frac{\lambda_i}{d}\right)^s, \tag{12}$$

where λ_i is the eigenvalue associated to the elementary function ϕ_i. As a consequence, the value $r(1)$ is:

$$r(1) = \frac{\sum_{i \neq 0} a_i^2 \left(1 - \frac{\lambda_i}{d}\right)}{\sum_{j \neq 0} a_j^2} = 1 - \frac{\sum_{i \neq 0} a_i^2 \frac{\lambda_i}{d}}{\sum_{j \neq 0} a_j^2},$$

and the autocorrelation coefficient can be computed as

$$\xi = \frac{d \sum_{j \neq 0} a_j^2}{\sum_{i \neq 0} a_i^2 \lambda_i}. \tag{13}$$

The sum of the squared Fourier coefficients a_j^2 associated to the same eigenvalue λ_i is $|X|(\overline{f_i^2} - \overline{f_i}^2)$, where f_i is the sum of all the elementary components $a_i \phi_i$ with the same eigenvalue λ_i and the overline represents the average over the entire search space X. The sum of the squared Fourier

coefficients a_j^2 with $j \neq 0$ is $|X|(\overline{f^2} - \overline{f}^2)$. Introducing these two expressions in (13) we can write:

$$\xi = \left(\frac{\sum_{i \neq 0} (\overline{f_i^2} - \overline{f_i}^2)\lambda_i}{d(\overline{f^2} - \overline{f}^2)} \right)^{-1} = \left(\sum_{i \neq 0} W_i \frac{\lambda_i}{d} \right)^{-1},$$

where the values W_i are called spectral coefficients and are defined as

$$W_i = \frac{\overline{f_i^2} - \overline{f_i}^2}{\overline{f^2} - \overline{f}^2}. \quad (14)$$

The autocorrelation length ℓ can also be expressed with the help of (12) as:

$$\ell = \sum_{s=0}^{\infty} r(s) = d \sum_{i \neq 0} \frac{W_i}{\lambda_i}. \quad (15)$$

Thus, the problem of computing the autocorrelation coefficient and length is reduced to the problem of finding the spectral coefficients W_i. These coefficients have been computed for some NP-hard combinatorial optimization problems. Chicano et al. [10] provide expressions for the autocorrelation measures in the case of QAP. They do not provide any algorithm to do it in the paper but they implemented an $O(n^2)$ algorithm to compute the autocorrelation measures which is available on-line(see Section 6). For illustration purposes we show here the values for autocorrelation measures and spectral coefficient of the Unconstrained Quadratic Optimization (UQO). The details can be found in [6]. The autocorrelation coefficient and length is given by the following expressions:

$$\xi = \frac{n}{2(2 - W_2)}, \quad (16)$$

$$\ell = \frac{n(1 + W_2)}{4}, \quad (17)$$

where n is the size of the problem (length of the binary solutions) and W_2 is given by:

$$W_2 = \frac{\overline{f_2^2}}{\overline{f^2} - \overline{f}^2}. \quad (18)$$

The expressions for $\overline{f_2^2}$, $\overline{f^2}$ and $\overline{f}^2$ are:

$$\overline{f}^2 = \frac{1}{16} \left(\sum_{i,j=1}^{n} q_{ij} + \sum_{i=1}^{n} q_{ii} \right)^2, \quad (19)$$

$$\overline{f_2^2} = \frac{1}{16} \sum_{i=1}^{n} v_i^2, \quad (20)$$

$$\overline{f^2} = \sum_{i,j=1}^{n} \sum_{i',j'=1}^{n} \frac{q_{ij} q_{i'j'}}{2^{|\{i,j,i',j'\}|}}. \quad (21)$$

The previous expressions can be computed at most in $O(n^4)$. Sutton et al. [22] showed how the autocorrelation function $r(s)$ can be computed for all the pseudo-Boolean functions using the one-flip neighborhood. The previous expression is just a particular case of the Sutton's expression. In their work they provided expressions for the MAX-3-SAT problem.

4. FITNESS-DISTANCE CORRELATION

The Fitness-Distance Correlation (FDC) is a measure introduced by Jones and Forrest [15] to measure problem difficulty. Given all the solutions in the search space, it computes the correlation coefficient between the fitness values of these solutions and the Hamming distances of the solutions to their nearest global optimum. In the case of an optimization problem defined over a binary solution space we can define FDC as follows.

DEFINITION 2. *Given a function $f : \mathbb{B}^n \mapsto \mathbb{R}$ the fitness-distance correlation for f is defined as*

$$r = \frac{Cov_{fd}}{\sigma_f \sigma_d}, \quad (22)$$

where Cov_{fd} is the covariance of the fitness values and the distances of the solutions to their nearest global optimum, σ_f is the standard deviation of the fitness values in the search space and σ_d is the standard deviation of the distances to the nearest global optimum in the search space. Formally:

$$Cov_{fd} = \frac{1}{2^n} \sum_{x \in \mathbb{B}^n} (f(x) - \overline{f})(d(x) - \overline{d}),$$

$$\overline{f} = \frac{1}{2^n} \sum_{x \in \mathbb{B}^n} f(x), \quad \sigma_f = \sqrt{\frac{1}{2^n} \sum_{x \in \mathbb{B}^n} (f(x) - \overline{f})^2},$$

$$\overline{d} = \frac{1}{2^n} \sum_{x \in \mathbb{B}^n} d(x), \quad \sigma_d = \sqrt{\frac{1}{2^n} \sum_{x \in \mathbb{B}^n} (d(x) - \overline{d})^2}, \quad (23)$$

where the function $d(x)$ is the Hamming distance between x and its nearest global optimum.

The FDC r is a value between -1 and 1. The lower the absolute value of r, the more difficult the optimization problem is supposed to be. The exact computation of the FDC using the previous definition requires the evaluation of the complete search space. It is required to determine the global optima to define $d(x)$ and compute the statistics for d and f. If the objective function f is a constant function, then the FDC is not well-defined, since $\sigma_f = 0$. The next theorem, extracted from [8], provides an exact expression for FDC in the case in which there exists one only global optimum x^* and we know the elementary landscape decomposition of f.

THEOREM 1. *Let f be an objective function whose elementary landscape decomposition is $f = \sum_{p=0}^{n} f_{2p}$, where f_0 is the constant function $f_0(x) = \overline{f}$ and each f_{2p} with $p > 0$ is an order-p elementary function with zero offset. If there exists only one global optimum in the search space x^*, the FDC can be exactly computed as:*

$$r = \frac{-f_2(x^*)}{\sigma_f \sqrt{n}}. \quad (24)$$

The previous theorem shows that the only thing we need to know on the global optimum is the value of the first elementary component. With this information we can exactly compute the FDC. Some problems for which we know the elementary landscape decomposition based on the numerical data defining a problem instance are MAX-SAT, 0-1 Unconstrained Quadratic Optimization (UQO), the Subset Sum problem (SS), the NK-landscapes, etc. For all of them we could provide expressions for their FDC.

The result of the previous theorem starts an interesting discussion. In Section 3 we mentioned that some works on landscape analysis claim that the ruggedness of a landscape is related to its hardness [3]. In particular, the autocorrelation coefficient ξ and the autocorrelation length ℓ of a problem are two measures proposed to characterize an objective function in a way that allows one to estimate the performance of a local search method. Also a relationship has been noticed between the autocorrelation length and the expected number of local optima of a problem [12], in agreement with the autocorrelation length conjecture. In summary, empirical and theoretical results support the hypothesis that a rugged landscape is more difficult than a problem with a smooth landscape.

In the case of the elementary functions defined over binary strings, the functions with higher order are more rugged than the ones with lower order. The order-1 elementary landscapes are the smoothest landscapes and, in fact, they can always be solved in polynomial time. Following this chain of reasoning, in a general landscape, the elementary components with order $p > 1$ are the ones that make the problem difficult. However, from Theorem 1 we observe that only the order-1 elementary component of a function f is taken into account in the computation of the FDC. This fact poses some doubts on the value of the FDC as a measure of difficulty of a problem, since FDC is shown to neglect the rest of information captured in the higher order components. This is true under the assumption that one single global optimum exists in the search space. The doubts on FDC as being a difficulty indicator have also been raised by other authors. Two examples are the work by Tomassini et al. [23] focused on genetic programming and the one by Bierwirth et al. [4] based on the Job Shop Scheduling.

If the objective function is elementary, then the expression of the exact FDC is specially simple:

$$r = \begin{cases} \frac{\overline{f} - f(x^*)}{\sigma_f \sqrt{n}} & \text{if } p = 1, \\ 0 & \text{if } p > 1. \end{cases} \tag{25}$$

The previous equation states that only elementary landscapes with order $p = 1$ have a nonzero FDC. Furthermore, the FDC does depend on the value of the objective function in the global optimum $f(x^*)$ and the average value $\overline{f}$, but not on the solution x^* itself. We can also observe that if we are maximizing, then $f(x^*) > \overline{f}$ and the FDC is negative, while if we are minimizing $f(x^*) < \overline{f}$ and the FDC is positive. The order-1 elementary landscapes can always be written as linear functions and they can be optimized in polynomial time. That is, if f is an order-1 elementary function then it can be written in the following way:

$$f(x) = \sum_{i=1}^{n} a_i x_i + b. \tag{26}$$

where a_i and b are real values. If such a linear function has only one global optimum (that is, $a_i \neq 0$ for all i). The FDC can be computed using the expression:

$$r = \frac{-\sum_{i=1}^{n} |a_i|}{\sqrt{n \sum_{i=1}^{n} a_i^2}}, \tag{27}$$

which is always in the interval $-1 \leq r < 0$. When all the values of a_i are the same, the FDC computed with (27) is -1. This happens in particular for the Onemax problem. But if there exist different values for a_i, then we can reach any arbitrary value in $[-1, 0)$ for r. The next result provides a way to design a linear function with the value of r we want.

THEOREM 2. *Let ρ be an arbitrary real value in the interval $[-1, 0)$, then any linear function $f(x)$ given by (26) where $n > 1/\rho^2$, $a_2 = a_3 = \ldots = a_n = 1$ and a_1 is*

$$a_1 = \frac{(n-1) + n|\rho|\sqrt{(1-\rho^2)(n-1)}}{n\rho^2 - 1}, \tag{28}$$

has exactly FDC $r = \rho$.

Theorem 2, also extracted from [8] provides a solid argument against the use of FDC as a measure of the difficulty of a problem. In effect, we can always build an optimization problem based on a linear function, which can be solved in polynomial time, with an FDC as near as desired to 0 (but not zero), that is, as "difficult" as desired according to the FDC. However, we have to highlight here that for a given FDC value ρ we need at least $n > 1/\rho^2$ variables. Thus, an FDC nearer to 0 requires more variables.

Let us discuss now how can we use (24) in order to compute FDC for a given problem. Observe that we need the global optimum but most probably we don't know that global optimum (that's the reason we are interested in solving that problem). This is a drawback not only of our expression, but of any other procedure to compute FDC. In all the cases we need the global optima. Thus, we cannot compute the exact FDC in general, but an approximation. One empirical procedure to compute FDC would consist in sampling the search space, evaluating the fitness in all these solutions, computing the distances between any pair of solutions and, finally, computing FDC using (22). This procedure requires to store all the sampled points in memory until we decide which one is the optimum and we compute all the distances between them. In this case our expression (24) have some advantages to simplify the procedure. First, we don't need to store all the points in memory, just the optimal one. Thus, the sampling can be just a trajectory-based strategy (guided or not) that tries to find the global optimum. Second, once we have a solution x^* that will be used in the computation of FDC, we have just to evaluate f_2 on it and apply (24) to get r. Third, the FDC value computed in this way takes into account not only the sampled points, but all the points in the search space, that is, it is a statistic that considers the entire search space. Thus, we would expect it to be more accurate than the empirical FDC computed using only the sampled values. We have, however, to clarify that we need to compute σ_f, which in most of the cases is easy to do it from the problem data (UQO or QAP for example), but in some other problems it could be difficult or even intractable.

5. EXPECTED FITNESS IN MUTATION

In the previous sections we have seen how landscape theory can be used to efficiently compute autocorrelation measures or the FDC of a problem. All these measures have been considered in the past, not without criticisms, as indicators of problem hardness. In this section we show that in the case of optimization problems with a binary solution space, the elementary landscape decomposition can also be used to compute the expected fitness value of a solution when it is mutated using bit-flip mutation. Furthermore, we will see that there is a correspondence between the expectation curves (expectation against probability of flipping

a bit) and the number of components and the exact values of the elementary landscape decomposition.

First of all, let us define the bit-flip mutation operator. Given a solution x of size n (binary string), the bit-flip mutation operator changes the value of each bit with probability p, the only parameter of the mutation. If $0 < p < 1$ the mutation operator can yield any solution of the search space with a different probability. In particular, if we apply the operator to solution x with probability p of flipping a bit, then the solution y will be obtained with probability $Prob\{y = M_p(x)\} = p^{|x \oplus y|}(1-p)^{n-|x \oplus y|}$, where $\oplus$ denotes the exclusive OR and $|\cdot|$ is the function that counts the number ones in a string.

Sutton et al. [21] and Chicano et al. [7] discovered that the expected value of the fitness of a solution x after a bit-flip can be easily computed using the elementary landscape decomposition. The result is the following (extracted from [7]):

THEOREM 3. *Let f be an arbitrary function whose elementary decomposition in the one-change binary configuration space is:*

$$f = \sum_{j=1}^{n} f_{2j}, \tag{29}$$

where f_{2j} denotes the order-j elementary component (with eigenvalue $2j$). Then the expected fitness value of a solution after the application of the bit-flip mutation operator to x is:

$$\mathbb{E}[f(M_p(x))] = \overline{f} + \sum_{j=1}^{n}(1-2p)^j(f_{2j}(x) - \overline{f_{2j}}), \tag{30}$$

where $\overline{f_{2j}}$ denotes the average value of function $f_{2j}(x)$ in the entire search space.

With this expression we can efficiently compute the expected value after the application of the bit-flip mutation operator to solution x for an arbitrary function f. The complexity of this operation is the sum of the complexities of the evaluation of the elementary components. The average value $\overline{f_{2j}}$ is a constant that depends on the parameters of the particular instance we are solving and can be precomputed before the search process.

It is our experience that usually we can find an algorithm for computing the component f_{2j} and the value $\overline{f_{2j}}$ that has the same complexity as the original function f. If this is true for a problem the complexity of computing $\mathbb{E}[f(M_p(x))]$ is at most n times the complexity of computing a particular component f_{2j}. One interesting observation is that in many problems the number of elementary components is a fixed number lower than n, independently of the instance. For example, in the MAX-k-SAT problem the objective function can be written as a sum of k elementary landscapes [16]. In these cases the computation of $\mathbb{E}[f(M_p(x))]$ will have the same complexity as the computation of the hardest elementary component f_{2j}.

The curves of $\mathbb{E}[f(M_p(x))]$ (depending on p) for general functions f can be almost arbitrary. The only limitations are that they must be a polynomial of degree at most n and at $p = 1/2$ the value of $\mathbb{E}[f(M_p(x))]$ is always $\bar{f}$. We can state that an elementary component with order j (and eigenvalue $2j$) is related to a polynomial of degree j in the expectation curve. As an example we show in Figure 2 the curve $\mathbb{E}[f(M_p(x))]$ for a function which can be decomposed into three elementary landscapes. We show in the figure the expectation (solid line) and the contribution of each elementary component (dashed lines). We can observe in this case that the function value $f(x)$ is $\bar{f}$ (see the value for $p = 0$). However, in spite of this, the expectation does depend on p because we are not dealing with an elementary landscape. Furthermore, it has the maximum value at $p = (4 - \sqrt{7})/6 \approx 0.226$. We can observe that for $p = 1/2$ the expectation crosses $\bar{f}$ again.

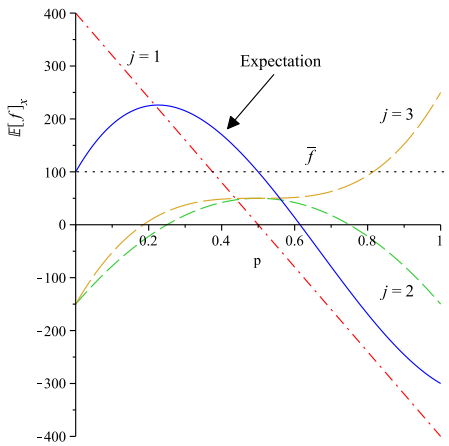

Figure 2: Expectation $\mathbb{E}[f(M_p(x))]$ for a function with three elementary components.

From a theoretical point of view the results presented in this section shed some light on the behaviour of the bit-flip mutation operator. This knowledge could be used, for example, in the theoretical analysis of the runtime of search algorithms in which this operator is used.

At this point we can observe that, given a solution x, the curve providing the expected fitness after mutating x as a function of p is completely determined by the values of the elementary components of f in x. This means, that we can determine the number and the values of the elementary components in x just analyzing the bit-flip mutation operator. In this sense, there is a one-to-one correspondence between the behaviour of the bit-flip mutation in x and the elementary components in x. If we consider all the solutions in the search space, we conclude that it is not only the case that we can determine the behaviour of the bit-flip mutation operator using the elementary landscape decomposition of the problem, but we can extract the elementary landscape decomposition of the problem analyzing the bit-flip mutation.

Previous work has considered using the expectation curves to compute the probability value p^* optimizing the expected behaviour. It was argued in [7], for example, that this could be the base for a new operator that could escape faster from non-optimal regions. However, using the probability p^* for which the expected fitness is optimal does not mean that the algorithm will find the global optimum faster, we should consider the whole search process. Studies still under development prove that landscape theory is also useful when we want to compute the expected number of evaluations (or generations) required to find the global optimum in $(1 + \lambda)$ EAs. This opens an encouraging line of research focused on the link between the elementary landscape decomposition of combinatorial optimization problems and an objective mea-

Instance	ξ	ℓ	Avg_{sd} time (ms)
sko100a	27.7997	29.9852	492_{285}
sko100b	28.1060	30.4703	530_{290}
sko100c	27.5476	29.5778	519_{294}
sko100d	27.5351	29.5573	531_{283}
sko100e	27.6002	29.6634	513_{297}
sko100f	27.3459	29.2465	521_{293}
tai100a	25.1950	25.3830	523_{286}
tai100b	35.4719	39.6132	512_{290}
wil100	28.3622	30.8679	539_{282}
esc128	32.0000	32.0000	505_{308}
tho150	41.1901	44.1743	517_{289}
tai150b	40.4581	42.9472	521_{282}
tai256c	64.0000	64.0000	602_{287}

Table 1: Autocorrelation coefficient ξ and length ℓ of the larger instances of QAPLIB. We also show the average and standard deviation of the time required to compute the measures.

sure of problem hardness such as the time required to solve it using a concrete algorithm.

6. SOFTWARE TOOLS

Researchers working with landscape theory frequently implement some small code snippets and algorithms to support their research. Among the software implemented we can find applications to compute the autocorrelation measures of optimization problems, assist the researcher in the task of finding the elementary landscape decomposition of a problem, empirically check if a landscape is elementary, apply some of the ideas in landscape theory in order to improve a search strategy and so on. In most of the cases these algorithms are not published by the researcher, since it is considered a tool for supporting the research, but not a product. These algorithms are usually far from trivial to implement because they require a deep knowledge of the mathematical details of landscape theory.

However, researchers interested in problem understanding would like sometimes to have such software tools as the base for further increasing their knowledge on a problem without having to worry about the implementation details of the algorithms. In this section we describe two of such software tools that are available for the researchers.

The first one is a small Web application that computes the autocorrelation measures for QAP instances. The user can select an instance of the QAPLIB or can upload her/his own QAP instance in a file with the same format as the instances in QAPLIB. As a result the application computes the spectral coefficients, the autocorrelation coefficient and the autocorrelation length of the instance. The algorithm used to compute these values is a sophisticated algorithm of order $O(n^2)$. This Web application is available at URL `http://neo.lcc.uma.es/software/qap.php`. In Table 6 we show the autocorrelation coefficient and length of the larger instances of QAPLIB computed with the tool. The average and standard deviation of the time required was computed over 100 independent runs.

The second tool we will describe is a desktop tool with a GUI that was designed to be a software lab for landscape theory research. Its name is *Landscape Explorer*, available at `http://neo.lcc.uma.es/software/landexplorer` (Figure 3). The two main requirements we took into account in the design of the tool were the extensibility and the multi-platform support. For this reason we decided to base our tool on the *Rich Client Platform* (RCP) of *Eclipse*, using Java as programming language. We can think in *Landscape Explorer* as a collection of *plug-ins* with dependencies among them. In order to extend the application with a new feature we just need to build a plug-in implementing that feature. This tool can be extended to include new landscapes or procedures for landscape analysis. In order to add a new landscape the developer has just to provide an implementation of the neighborhood and the objective function.

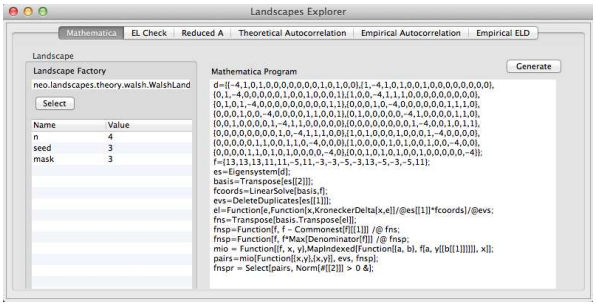

Figure 3: Screen capture of Landscape Explorer.

At this moment Landscape Explorer can work with the following landscapes: Quadratic Assignment Problem (QAP), Traveling Salesman Problem (TSP), Unconstrained Quadratic Optimization (UQO), Subset Sum (SS), Frequency Assignment Problem (FAP), DNA Fragment Assembly (DFA), linear combinations of Walsh Functions. The last landscape is not a combinatorial optimization problem, but any optimization problem defined over binary strings can be expressed as a linear combination of Walsh functions. Thus, it is a convenient landscape to model any problem in binary strings.

Regarding the procedures implemented in Landscape Explorer we find the following ones:

- **Empirical Autocorrelation Computation**. This method performs a random walk over the search space jumping from one solution x to one of its neighbors $y \in N(x)$. At the same time it computes the autocorrelation of the fitness values and finally shows the Weinberger autocorrelation function, the autocorrelation length and the autocorrelation coefficient.
- **Elementary Landscape Check**. This procedure samples the search space and their neighbors in order to check whether the landscape is elementary or not. This method is not exhaustive: if the answer is "yes" the user cannot be sure that the landscape is elementary. On the other hand, if we *a priori* know that the landscape is elementary, this procedure can be used to obtain the eigenvalue and the offset.
- ***Mathematica* program**. Given a landscape this procedure generates a Mathematica script to find the elementary landscape decomposition. This procedure is a key part of the methodology to find the elementary landscape decomposition of a problem [11].
- **Estimation of the number of elementary components**. Given a landscape defined over binary strings, this procedure empirically determines the number of elementary components of the objective function.

All the previous procedures can be applied to any landscape implemented in the application. In addition to these procedures, Landscape Explorer includes some algorithms to exactly compute autocorrelation measures for some specific problems: QAP, TSP, UQO and SS.

7. CONCLUSIONS AND FUTURE WORK

Landscape theory is a convenient framework to understand optimization problems. We have shown through the paper how can we use landscape theory to efficiently compute statistics and measures of optimization problems. In particular, we showed that autocorrelation measures, fitness-distance correlation and the expected fitness after a bit-flip mutation can be efficiently computed from problem data. In the context of problem understanding these and other measures are useful to get a deep knowledge on the optimization problems. We also described some software tools implementing algorithms that can compute the measures and statistics.

We think landscape theory can be used to find new statistics of the problems. In particular, we have some preliminary results that suggest a link between runtime analysis and landscape theory. We plan to exploit this link in order to predict the behaviour of the algorithms without executing them. Such predictions would allow one to select the optimal parameters to run an algorithm before it is executed.

8. ACKNOWLEDGEMENTS

This research has been partially funded by the Spanish Ministry of Economy and Competitiveness and FEDER under contract TIN2011-28194 (the roadME project).

9. REFERENCES

[1] E. Angel and V. Zissimopoulos. Autocorrelation coefficient for the graph bipartitioning problem. *Theoretical Compuer Science*, 191:229–243, 1998.

[2] E. Angel and V. Zissimopoulos. On the landscape ruggedness of the quadratic assignment problem. *Theoretical Computer Sciences*, 263:159–172, 2000.

[3] J. W. Barnes, B. Dimova, and S. P. Dokov. The theory of elementary landscapes. *Applied Mathematics Letters*, 16:337–343, 2003.

[4] C. Bierwirth, D. Mattfeld, and J.-P. Watson. Landscape regularity and random walks for the job-shop scheduling problem. In *EvoCOP*, LNCS 3004, pages 21–30. 2004.

[5] T. Biyikoglu, J. Leyold, and P. F. Stadler. *Laplacian Eigenvectors of Graphs*. Lecture Notes in Mathematics. Springer-Verlag, 2007.

[6] F. Chicano and E. Alba. Elementary landscape decomposition of the 0-1 unconstrained quadratic optimization. *Journal of Heuristics*. DOI:10.1007/s10732-011-9170-6.

[7] F. Chicano and E. Alba. Exact computation of the expectation curves of the bit-flip mutation using landscapes theory. In *Proceedings of GECCO*, pages 2027–2034, 2011.

[8] F. Chicano and E. Alba. Exact computation of the fitness-distance correlation for pseudoboolean functions with one global optimum. In *Proceedings of EvoCOP*, volume 7245 of *LNCS*, pages 111–123. Springer, 2012.

[9] F. Chicano, F. Daolio, G. Ochoa, S. Vérel, M. Tomassini, and E. Alba. Local optima networks, landscape autocorrelation and heuristic search performance. In *Proceedings of the PPSN*, volume 7492 of *LNCS*, pages 337–347. Springer, 2012.

[10] F. Chicano, G. Luque, and E. Alba. Autocorrelation measures for the quadratic assignment problem. *Applied Mathematics Letters*, 25(4):698–705, 2012.

[11] F. Chicano, L. D. Whitley, and E. Alba. A methodology to find the elementary landscape decomposition of combinatorial optimization problems. *Evol. Computation*, 19(4):597–637, 2011.

[12] R. García-Pelayo and P. Stadler. Correlation length, isotropy and meta-stable states. *Physica D: Nonlinear Phenomena*, 107(2-4):240–254, Sep 1997.

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

[14] L. K. Grover. Local search and the local structure of NP-complete problems. *Operations Research Letters*, 12:235–243, 1992.

[15] T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *GECCO*, pages 184–192. Morgan Kaufmann, 1995.

[16] S. Rana, R. B. Heckendorn, and D. Whitley. A tractable walsh analysis of SAT and its implications for genetic algorithms. In *Proceedings of AAAI*, pages 392–397, Menlo Park, CA, USA, 1998.

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

[18] P. F. Stadler. Toward a theory of landscapes. In R. L.-P. et al., editor, *Complex Systems and Binary Networks*, pages 77–163. Springer-Verlag, 1995.

[19] P. F. Stadler. Landscapes and their correlation functions. *Journal of Mathematical Chemistry*, 20:1–45, 1996.

[20] P. F. Stadler. *Biological Evolution and Statistical Physics*, chapter Fitness Landscapes, pages 183–204. Springer, 2002.

[21] A. M. Sutton, D. Whitley, and A. E. Howe. Mutation rates of the (1+1)-EA on pseudo-boolean functions of bounded epistasis. In *Proceedings of GECCO*, pages 973–980, 2011.

[22] A. M. Sutton, L. D. Whitley, and A. E. Howe. A polynomial time computation of the exact correlation structure of k-satisfiability landscapes. In *Proceedings of GECCO*, pages 365–372, 2009.

[23] M. Tomassini, L. Vanneschi, P. Collard, and M. Clergue. A study of fitness distance correlation as a difficulty measure in genetic programming. *Evolutionary Computation*, 13(2):213–239, 2005.

[24] E. Weinberger. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological Cybernetics*, 63(5):325–336, 1990.

[25] D. Whitley, A. M. Sutton, and A. E. Howe. Understanding elementary landscapes. In *Proceedings of GECCO*, pages 585–592. ACM, 2008.

[26] L. D. Whitley and A. M. Sutton. Partial neighborhoods of elementary landscapes. In *Proceedings of GECCO*, pages 381–388. ACM, 2009.