Elementary Landscape Decomposition of the Test Suite Minimization Problem

Francisco Chicano, Javier Ferrer, and Enrique Alba

University of Málaga, Spain,
{chicano,ferrer,alba}@lcc.uma.es

Abstract. Landscape theory provides a formal framework in which combinatorial optimization problems can be theoretically characterized as a sum of a special kind of landscape called elementary landscape. The decomposition of the objective function of a problem into its elementary components provides additional knowledge on the problem that can be exploited to create new search methods for the problem. We analyze the Test Suite Minimization problem in Regression Testing from the point of view of landscape theory. We find the elementary landscape decomposition of the problem and propose a practical application of such decomposition for the search.

Keywords: Fitness landscapes, test suite minimization, regression testing, elementary landscapes

1 Introduction

The theory of landscapes focuses on the analysis of the structure of the search space that is induced by the combined influences of the objective function of the optimization problem and the choice neighborhood operator [8]. In the field of combinatorial optimization, this theory has been used to characterize optimization problems and to obtain global statistics of the problems [11]. However, in recent years, researchers have been interested in the applications of landscape theory to improve the search algorithms [5].

A *landscape* for a combinatorial optimization problem is a triple (X, N, f), where $f : X \mapsto \mathbb{R}$ defines the objective function and the *neighborhood operator* function $N(x)$ generates the set of points reachable from $x \in X$ in a single application of the neighborhood operator. If $y \in N(x)$ then y is a neighbor of x.

There exists a special kind of landscapes, called *elementary landscapes*, which are of particular interest due to their properties [12]. We define and analyze the elementary landscapes in Section 2, but we can advance that they are characterized by the *Grover's wave equation*:

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

where d is the size of the neighborhood, $|N(x)|$, which we assume is the same for all the solutions in the search space, $\bar{f}$ is the average solution evaluation over

the entire search space, λ is a characteristic constant and $\text{avg}\{f(y)\}_{y \in N(x)}$ is the average of the objective function f computed in its neighborhood:

$$\underset{y \in N(x)}{\text{avg}}\{f(y)\} = \frac{1}{|N(x)|} \sum_{y \in N(x)} f(y) \tag{1}$$

For a given problem instance whose objective function is elementary, the values $\bar{f}$ and λ can be easily computed in an efficient way, usually from the problem data. Thus, 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 $f(x)$, without evaluating any of the neighbors. This means that in elementary landscapes we get additional information from a single solution evaluation. We get an idea of what is the quality of the solutions around the current one. This information can be used to design more clever search strategies and operators which effectively use the information.

Lu *et al.* [5] provide a nice example of the application of the landscape analysis to improve the performance of a search method. In their work, the performance of the Sampling Hill Climbing is improved by avoiding the evaluation of non-promising solutions. The average fitness value in the neighborhood of the solutions computed with (1) is at the core of their proposal.

When the landscape is not elementary it is always possible to write the objective function as a sum of elementary components, called *elementary landscape decomposition* of a problem [1]. Then, Grover's wave equation can be applied to each elementary component and all the results are summed to give the average fitness in the neighborhood of a solution. Furthermore, for some problems the average cannot be limited to the neighborhood of a solution, but it can be extended to the second-order neighrbors (neighbors of neighbors), third-order neighbors, and, in general, to any arbitrary region around a given solution, including the whole search space. Sutton *et al.* [10] show how to compute the averages over *spheres* and *balls* of arbitrary radius around a given solution in polynomial time using the elementary landscape decomposition of real-valued functions over binary strings. In [9] they propose a method that uses these averages over the balls around a solution to escape from plateaus in the MAX-k-SAT problem. The empirical results noticed an improvement when the method was applied. Langdon [4] also analyzed the spheres of arbitrary radius from the point of view of landscape theory, highlighting that the Walsh functions are eigenvectors of the spheres and the mutation matrix in GAs.

If we extend the landscape analysis of the objective function f to their powers (f^2, f^3, etc.), Grover's wave equation allows one to compute higher-order moments of the fitness distribution around a solution and, with them, the variance, the skewness and the kurtosis of this distribution. Sutton *et al.* [10] provide an algorithm for this computation.

We analyze here the Test Suite Minimization problem in regression testing from the point of view of landscape theory. This software engineering problem consists in selecting a set of test cases from a large test suite that satisfies a given condition, like maximizing the coverage and minimizing the oracle cost [13].

The remainder of the paper is organized as follows. In Section 2 we present the mathematical tools required to understand the rest of the paper and Section 3 formally defines the Test Suite Minimization problem. Section 4 presents the two main contributions: the elementary landscape decomposition of the objective function of the problem and its square. We provide closed-form formulas for both f and f^2. In the mathematical development we include a novel application of the Krawtchouk matrices to the landscape analysis. Section 5 proposes an application of the decompositions of f and f^2 and presents a short experimental study showing the benefits (and drawbacks) of the proposal. Finally, with Section 6 we conclude the paper.

2 Background

In this section we present some fundamental results of landscape theory. We will only focus on the relevant information required to understand the rest of the paper. The interested reader can deepen on this topic in [7].

Let (X, N, f) be a landscape, where X is a finite set of solutions, $f : X \to \mathbb{R}$ is a real-valued function defined on X and $N : X \to \mathcal{P}(X)$ is the neighborhood operator. The adjacency and degree matrices of the neighborhood N are defined as:

$$A_{xy} = \begin{cases} 1 \text{ if } y \in N(x) \\ 0 \text{ otherwise} \end{cases} ; \quad D_{xy} = \begin{cases} |N(x)| \text{ if } x = y \\ 0 \qquad \text{otherwise} \end{cases} \tag{2}$$

We restrict our attention to regular neighborhoods, where $|N(x)| = d > 0$ for a constant d, for all $x \in X$. Then, the degree matrix is $D = dI$, where I is the identity matrix. The Laplacian matrix Δ associated to the neighborhood is defined by $\Delta = A - D$. In the case of regular neighborhoods it is $\Delta = A - dI$. 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 A acts on function f as follows

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

Thus, the component x of $(A\ f)$ is the sum of the function value of all the neighbors of x. Stadler defines the class of *elementary landscapes* where the function f is an eigenvector (or eigenfunction) of the Laplacian up to an additive constant [8]. Formally, we have the following

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

We use $-\Delta$ instead of Δ in the definition to avoid negative eigenvalues. In connected neighborhoods (the ones we consider here) the offset b is the average value of the function over the whole search space: $b = \bar{f}$. Taking into account basic results of linear algebra, it can be proved that if f is elementary with eigenvalue λ, $af + b$ is also elementary with the same eigenvalue λ. Furthermore, in regular neighborhoods, if g is an eigenfunction of $-\Delta$ with eigenvalue λ then g is also an eigenvalue of 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 $\text{avg}\{f(y)\}_{y \in N(x)} = \frac{1}{d}(A\ f)(x)$. If f is an elementary function with eigenvalue λ, then the average is computed as:

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

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

The previous definitions are general concepts of landscape theory. Let us focus now on the binary strings with the one-change neighborhood, which is the representation and the neighborhood we use in the test suite minimization problem. In this case the solution set X is the set of all binary strings of size n. Two solutions x and y are neighboring if one can be obtained from the other by flipping a bit, that is, if the Hamming distance between the solutions, denoted with $\mathcal{H}(x, y)$, is 1. We define the sphere of radius k around a solution x as the set of all solutions lying at Hamming distance k from x [10]. A *ball* of radius k is the set of all the solutions lying at Hamming distance lower or equal to k. In analogy to the adjacency matrix we define the sphere and ball matrices of radius k as:

$$S_{xy}^{(k)} = \begin{cases} 1 \text{ if } \mathcal{H}(x, y) = k \\ 0 \text{ otherwise} \end{cases}; \qquad B_{xy}^{(k)} = \sum_{\rho=0}^{k} S_{xy}^{(\rho)} = \begin{cases} 1 \text{ if } \mathcal{H}(x, y) \leq k \\ 0 \text{ otherwise} \end{cases} \tag{4}$$

Since the ball matrices are based on the sphere matrices we can focus on the latter. The sphere matrix of radius one is the adjacency matrix of the one-change neighborhood, A, and the sphere matrix of radius zero is the identity matrix, I. Following [10], the matrices $S^{(k)}$ can be defined using the recurrence:

$$S^{(0)} = I; \quad S^{(1)} = A; \quad S^{(k+1)} = \frac{1}{k+1}\left(A \cdot S^{(k)} - (n - k + 1)S^{(k-1)}\right) \tag{5}$$

With the help of the recurrence we can write all the matrices $S^{(k)}$ as polynomials in A, the adjacency matrix. For example, $S^{(2)} = \frac{1}{2}\left(A^2 - nI\right)$. As we previously noted, the eigenvectors of the Laplacian matrix Δ are eigenvectors of the adjacency matrix A. On the other hand, if f is eigenvector of A, then it is also an eigenvector of any polynomial in A. As a consequence, all the functions that are elementary are eigenvectors (up to an additive constant) of $S^{(k)}$ and

their eigenvalues can be computed using the same polynomial in A that gives the expression for $S^{(k)}$. The same is true for the ball matrices $B^{(k)}$, since they are a sum of sphere matrices. Let us define the following series of polynomials:

$$S^{(0)}(x) = 1 \tag{6}$$

$$S^{(1)}(x) = x \tag{7}$$

$$S^{(k+1)}(x) = \frac{1}{k+1}\left(x \cdot S^{(k)}(x) - (n-k+1)S^{(k-1)}(x)\right) \tag{8}$$

We use the same name for the polynomials and the matrices related to the spheres. The reader should notice, however, that the polynomials will be always presented with their argument and the matrices have no argument. That is, $S^{(k)}$ is the matrix and $S^{(k)}(x)$ is the polynomial. Using the previous polynomials, the matrix $S^{(k)}$ can be written as $S^{(k)}(A)$ (the polynomial $S^{(k)}(x)$ evaluated in the matrix A) and any eigenvector g of A with eigenvalue λ is also an eigenvector of $S^{(k)}(A)$ with eigenvalue $S^{(k)}(\lambda)$.

One relevant set of eigenvectors of the Laplacian in the binary representation is that of Walsh functions [11]. Furthermore, the Walsh functions form an orthogonal basis of eigenvectors in the configuration space. Thus, they have been used to find the elementary landscape decomposition of problems with a binary representation like the SAT [6]. We will use these functions to provide the landscape decomposition of the objective function of the test suite minimization problem. Given the space of binary strings of length n, $\mathbb{B}^n$, a (non-normalized) Walsh function with parameter $w \in \mathbb{B}^n$ is defined as:

$$\psi_w(x) = \prod_{i=1}^{n}(-1)^{w_i x_i} = (-1)^{\sum_{i=1}^{n} w_i x_i} \tag{9}$$

Two useful properties of Walsh functions are $\psi_w \cdot \psi_v = \psi_{w+v}$ where $w+v$ is the bitwise sum in $\mathbb{Z}_2$ of w and v; and $\psi_w^2 = \psi_w \cdot \psi_w = \psi_{2w} = \psi_0 = 1$. We define the *order* of a Walsh function ψ_w as the value $\langle w|w\rangle = \sum_{i=1}^{n} w_i$, that is, the number of ones in w. A Walsh function with order p is elementary with eigenvalue $\lambda = 2p$ [8]. The average value of a Walsh function of order $p > 0$ is zero, that is, $\overline{\psi_w} = 0$ if w has at least one 1. The only Walsh function of order $p = 0$ is $\psi_0 = 1$, which is a constant.

In the mathematical development of Section 4 we will use, among others, Walsh functions of order 1 and 2. Thus, we present here a special compact notation for those binary strings having only one or two bits set to 1. We will denote with $\underline{i}$ the binary string with position i set to 1 and the rest set to 0. We also denote with $\underline{i,j}$ $(i \neq j)$ the binary string with positions i and j set to 1 and the rest to 0. We omit the length of the string n, but it will be clear from the context. For example, if we are considering binary strings in $\mathbb{B}^4$ we have $\underline{1} = 1000$ and $\underline{2,3} = 0110$. Using this notation we can write

$$\psi_{\underline{i}}(x) = (-1)^{x_i} = 1 - 2x_i \tag{10}$$

Given a set of binary strings W and a binary string u we denote with $W \wedge u$ the set of binary strings that can be computed as the bitwise AND of a string

in W and u, that is, $W \wedge u = \{w \wedge u | w \in W\}$. For example, $\mathbb{B}^4 \wedge 0101 = \{0000, 0001, 0100, 0101\}$.

Since the Walsh functions form an orthogonal basis of $\mathbb{R}^{2^n}$, any arbitrary pseudoboolean function can be written as a weighted sum of Walsh functions in the following way:

$$f = \sum_{w \in \mathbb{B}^n} a_w \psi_w \tag{11}$$

where the values a_w are called Walsh coefficients. We can group together the Walsh functions having the same order to find the elementary landscape decomposition of the function. That is:

$$f^{(p)} = \sum_{\substack{w \in \mathbb{B}^n \\ \langle w|w \rangle = p}} a_w \psi_w \tag{12}$$

where each $f^{(p)}$ is an elementary function with eigenvalue $2p$. The function f can be written as a sum of the $n+1$ elementary components, that is: $f = \sum_{p=0}^{n} f^{(p)}$. Thus, any function can be decomposed in a sum of at most n elementary landscapes, since we can add the constant value $f^{(0)}$ to any of the other elementary components.

Once we know that the possible eigenvalues of the elementary components of any function f are $2p$ with $0 \leq p \leq n$, we can compute the possible eigenvalues of the sphere matrices. Since the size of the neighborhood is $d = n$, we conclude that the only possible eigenvalues for the spheres are $S^{(k)}(n-2p)$ with $p \in \{0, 1, \ldots, n\}$. With the help of Eqs. (6) to (8) we can write a recurrence formula for the eigenvalues of the sphere matrices whose solution is $S^{(k)}(n-2p) = \mathcal{K}_{k,p}^{(n)}$, where $\mathcal{K}_{k,p}^{(n)}$ is the (k,p) element of the n-th Krawtchouk matrix [10], which is an $(n+1) \times (n+1)$ integer matrix. We will use Krawtchouk matrices to simplify the expressions and reduce the computation of the elementary components of the test suite minimization. The interested reader can deepen on Krawtchouk matrices in [3]. One important property of the Krawtchouk matrices that will be useful in Section 4 is:

$$(1+x)^{n-p}(1-x)^p = \sum_{k=0}^{n} x^k \mathcal{K}_{k,p}^{(n)} \tag{13}$$

Each component $f^{(p)}$ of the elementary landscape decomposition of f is an eigenfunction of the sphere matrix of radius r with eigenvalue $S^{(r)}(n-2p) = \mathcal{K}_{r,p}^{(n)}$. Thus, we can compute the average fitness value in a sphere of radius r around a solution x as:

$$\underset{y|\mathcal{H}(y,x)=r}{\text{avg}}\{f(y)\} = \binom{n}{r}^{-1} \sum_{p=0}^{n} \mathcal{K}_{r,p}^{(n)} f^{(p)}(x) \tag{14}$$

We can also compute the c-th moment of the function f in a sphere of radius r if we know the elementary landscape decomposition of f^c:

$$\mu_c = \underset{y|\mathcal{H}(y,x)=r}{\text{avg}}\{f^c(y)\} = \binom{n}{r}^{-1} \sum_{p=0}^{n} \mathcal{K}_{r,p}^{(n)} \left(f^c\right)^{(p)}(x) \tag{15}$$

3 Test Suite Minimization Problem

When a piece of software is modified, the new software is tested using some previous test cases in order to check if new errors were introduced. This check is known as *regression testing*. In [14] Yoo and Harman provide a very complete survey on search-based techniques for regression testing. They distinguish three different related problems: test suite minimization, test case selection and test case prioritization. The problem we face here is the test suite minimization [13]. We define the problem as follows. Let $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ be a set of tests for a program and let $\mathcal{M} = \{m_1, m_2, \ldots, m_k\}$ be a set of elements of the program that we want to cover with the tests. After running all the tests $\mathcal{T}$ we find that each test can cover several program elements. This information is stored in a matrix T that is defined as:

$$T_{ij} = \begin{cases} 1 \text{ if node } m_i \text{ is covered by test } t_j \\ 0 \text{ otherwise} \end{cases} \tag{16}$$

We define the coverage of a subset of tests $X \subseteq \mathcal{T}$ as:

$$coverage(X) = |\{i | \exists j \in X, T_{ij} = 1\}| \tag{17}$$

The problem consists in finding a subset $X \subseteq \mathcal{T}$ such that the coverage is maximized while the number of tests cases in the set $|X|$ is minimized. We can define the objective function of the problem as the weighted sum of the coverage and the number of tests. Thus, the objective function can be written as:

$$f(X) = coverage(X) - c \cdot |X| \tag{18}$$

where c is a constant that set the relative importance of the cost and coverage. It can be interpreted as the cost of a test measured in the same units as the benefit of a new covered element in the software. We assume here that all the elements in $\mathcal{M}$ to be covered have the same value for the user and the cost of testing one test in $\mathcal{T}$ is the same for all of them. We defer to future work the analysis of the objective function when this assumption is not true. Although the function proposed is a weighted sum, which simplifies the landscape analysis, non-linear functions can be also used and analyzed.

In the following we will use binary strings to represent the solutions of the problem. Thus, we introduce the decision variables $x_j \in \mathbb{B}$ for $1 \leq j \leq n$. The variable x_j is 1 if test t_j is included in the solution and 0 otherwise. With

this binary representation the coverage, the number of ones of a string and the objective function f can be written as:

$$coverage(x) = \sum_{i=1}^{k} \max_{j=1}^{n}\{T_{ij}x_j\}; \quad ones(x) = \sum_{j=1}^{n} x_j \tag{19}$$

$$f(x) = \sum_{i=1}^{k} \max_{j=1}^{n}\{T_{ij}x_j\} - c \cdot ones(x) \tag{20}$$

4 Elementary Landscape Decomposition

In this section we present two of the main contributions of this work: the elementary landscape decomposition of f and f^2. In order to simplify the equations let us introduce some notation. Let us define the sets $V_i = \{j|T_{ij} = 1\}$. V_i contains the indices of the tests which cover the element m_i. We also use in the following the term T_i to refer to the binary string composed of the elements of the i-th row of matrix T. T_i is a binary mask with 1s in the positions that appear in V_i.

4.1 Decomposition of f

The goal of this section is to find the Walsh decomposition of f. We first decompose the functions $coverage(x)$ and $ones(x)$ into elementary landscapes and then we combine the results. Let us start by analyzing the *coverage* function and, in particular, let us write the maximum in its definition as a weighted sum of Walsh functions with the help of (10).

$$\begin{aligned}\max_{j=1}^{n}\{T_{ij}x_j\} &= 1 - \prod_{j=1}^{n}(1 - T_{ij}x_j) = 1 - \prod_{j\in V_i}(1 - x_j) \\ &= 1 - \prod_{j\in V_i}\frac{1+\psi_{\underline{j}}(x)}{2} = 1 - 2^{-|V_i|}\prod_{j\in V_i}(1+\psi_{\underline{j}}(x))\end{aligned} \tag{21}$$

We can expand the product of Walsh functions in (21) using $\psi_u\psi_v = \psi_{u+v}$ to get the Walsh decomposition of $\max_{j=1}^{n}$.

$$\begin{aligned}\max_{j=1}^{n}\{T_{ij}x_j\} &= 1 - 2^{-|V_i|}\prod_{j\in V_i}(1+\psi_{\underline{j}}(x)) = 1 - 2^{-|V_i|}\sum_{W\in\mathcal{P}(V_i)}\prod_{j\in W}\psi_{\underline{j}}(x) \\ &= 1 - 2^{-|V_i|}\sum_{w\in\mathbb{B}^n\wedge T_i}\psi_w(x)\end{aligned} \tag{22}$$

Using the Walsh decomposition we can obtain that elementary landscape decomposition. The elementary components are the sums of weighted Walsh functions having the same order (number of ones in the string w). We can distinguish two cases: the constant elementary component (with order 0) and

the non-constant components. Then, the elementary landscape decomposition of $\max_{j=1}^{n}$ is:

$$\max_{j=1}^{n}\{T_{ij}x_j\}^{(0)} = 1 - \frac{1}{2^{|V_i|}} \tag{23}$$

$$\max_{j=1}^{n}\{T_{ij}x_j\}^{(p)} = -\frac{1}{2^{|V_i|}} \sum_{\substack{w \in \mathbb{B}^n \wedge T_i \\ \langle w,w\rangle = p}} \psi_w(x) \quad \text{where } p > 0 \tag{24}$$

Eqs. (23) and (24) are the elementary landscape decomposition of the coverage of one single software element. We just have to add all the components of all the k elements to get the elementary landscape decomposition of $coverage(x)$. However, we should highlight that the previous expression is not very efficient to compute the components of the maximum. We can observe that it requires to compute a sum of $\binom{|V_i|}{p}$ Walsh functions. Before combining all the pieces to get the elementary landscape decomposition of the objective function of the problem, we need first to find a simpler and more efficient expression for the elementary components of the coverage of one single element.

Up to the best of our knowledge, this is the first time that the following mathematical development is performed in the literature. The essence of the development, however, is useful by itself and can be applied to other problems with binary representation in which the Walsh analysis can be applied (like the Max-SAT problem). We will focus on the summation of (24). Let us rewrite this expression again as:

$$\sum_{\substack{w \in \mathbb{B}^n \wedge T_i \\ \langle w,w\rangle = p}} \psi_w(x) = \sum_{\substack{W \in \mathcal{P}(V_i) \\ |W| = p}} \prod_{j \in W} \psi_{\underline{j}}(x) \tag{25}$$

Now we can identify the second member of the previous expression with the coefficient of a polynomial. Let us consider the polynomial $Q_x^{(i)}(z)$ defined as:

$$Q_x^{(i)}(z) = \prod_{j \in V_i} (z + \psi_{\underline{j}}(x)) = \sum_{l=0}^{|V_i|} z^l \left(\sum_{\substack{W \in \mathcal{P}(V_i) \\ |W| = |V_i| - l}} \prod_{j \in W} \psi_{\underline{j}}(x) \right) = \sum_{l=0}^{|V_i|} q_l z^l \tag{26}$$

From (26) we conclude that the summation in (25) is the coefficient of $z^{|V_i|-p}$ in the polynomial $Q_x^{(i)}(z)$, that is, $q_{|V_i|-p}$. According to (10) and (26) we can write $Q_x^{(i)}(z) = (z+1)^{n_0^{(i)}}(z-1)^{n_1^{(i)}}$ where $n_0^{(i)}$ and $n_1^{(i)}$ are the number of zeros and ones, respectively, in the positions x_j of the solution with $j \in V_i$. It should be clear that $n_0^{(i)} + n_1^{(i)} = |V_i|$. Now we can profit from the fact that, according to (13), the polynomials $Q_x^{(i)}(z)$ are related to the Krawtchouk matrices by $Q_x^{(i)}(z) = (-1)^{n_1^{(i)}} \sum_{l=0}^{|V_i|} \mathcal{K}_{l,n_1^{(i)}}^{|V_i|} z^l$ and we can write $q_l = (-1)^{n_1^{(i)}} \mathcal{K}_{l,n_1^{(i)}}^{|V_i|}$. Finally

we obtain:

$$\sum_{\substack{w \in \mathbb{B}^n \wedge T_i \\ \langle w, w \rangle = p}} \psi_w(x) = \sum_{\substack{W \in \mathcal{P}(V_i) \\ |W| = p}} \prod_{j \in W} \psi_{\underline{j}}(x) = q_{|V_i|-p} = (-1)^{n_1^{(i)}} \mathcal{K}^{|V_i|}_{|V_i|-p, n_1^{(i)}} \quad (27)$$

The first N Krawtchouk matrices can be computed in $O(N^3)$. Furthermore, they can be computed once and stored in a file for future use. Thus, we transform the summation over a large number of Walsh functions into a count of the number of ones in a bit string and a read of a value stored in memory, which has complexity $O(n)$. Eq. (27) is an important result that allows us to provide an algorithm for evaluating the elementary landscape decomposition of our objective function. This algorithm is more efficient than the one proposed by Sutton *et al.* in [10]. We can now extend the elementary landscape decomposition to the complete coverage of all the elements. That is:

$$coverage^{(0)}(x) = \sum_{i=1}^{k} \max_{j=1}^{n} \{T_{ij} x_j\}^{(0)} = \sum_{i=1}^{k} \left(1 - \frac{1}{2^{|V_i|}}\right) \quad (28)$$

$$coverage^{(p)}(x) = \sum_{i=1}^{k} \max_{j=1}^{n} \{T_{ij} x_j\}^{(p)} = -\sum_{i=1}^{k} \frac{1}{2^{|V_i|}} (-1)^{n_1^{(i)}} \mathcal{K}^{|V_i|}_{|V_i|-p, n_1^{(i)}} \quad (29)$$

where $p > 0$. The previous expressions can be computed in $O(nk)$.

We now need the decomposition of the function $ones(x)$:

$$ones(x) = \sum_{j=1}^{n} x_j = \sum_{j=1}^{n} \frac{1 - \psi_{\underline{j}}(x)}{2} = \frac{n}{2} - \frac{1}{2} \sum_{j=1}^{n} \psi_{\underline{j}}(x) \quad (30)$$

Then, we can write:

$$ones^{(0)}(x) = \frac{n}{2}; \quad ones^{(1)}(x) = \frac{-1}{2} \sum_{j=1}^{n} \psi_{\underline{j}}(x) = ones(x) - \frac{n}{2} \quad (31)$$

which is the elementary landscape decomposition of $ones(x)$. Finally, we combine this result with the decomposition of $coverage(x)$ to obtain the decomposition of f:

$$f^{(0)}(x) = \sum_{i=1}^{k} \left(1 - \frac{1}{2^{|V_i|}}\right) - c \cdot \frac{n}{2} \quad (32)$$

$$f^{(1)}(x) = -\sum_{i=1}^{k} \frac{1}{2^{|V_i|}} (-1)^{n_1^{(i)}} \mathcal{K}^{|V_i|}_{|V_i|-1, n_1^{(i)}} - c \cdot \left(ones(x) - \frac{n}{2}\right) \quad (33)$$

$$f^{(p)}(x) = -\sum_{i=1}^{k} \frac{1}{2^{|V_i|}} (-1)^{n_1^{(i)}} \mathcal{K}^{|V_i|}_{|V_i|-p, n_1^{(i)}} \quad \text{where } 1 < p \leq n \quad (34)$$

All of the previous expressions can be computed in $O(nk)$. Since the maximum number of elementary components is equal to n, we can obtain the evaluation of all the elementary components of an arbitrary solution x in $O(n^2k)$. We found an algorithm with complexity $O(nk)$ to compute all the elementary components of f. This complexity is lower than the $O(n^n)$ complexity of the algorithm proposed in [10].

4.2 Decomposition of f^2

In the previous section we found the elementary landscape decomposition of f. In this section we are interested in the elementary landscape decomposition of f^2, since it allows to compute the variance in any region (sphere or ball) around any arbitrary solution x. The derivation of the elementary landscape decomposition of f^2 is based again in the Walsh analysis of the function. Combining the Walsh decomposition in (22) with the one of (30) and the definition of f in (20), the function f^2 can be written as:

$$f^2(x) = \left[\left(k - \frac{cn}{2}\right) - \sum_{i=1}^{k} \left(\frac{1}{2^{|V_i|}} \sum_{w \in \mathbb{B}^n \wedge T_i} \psi_w(x) \right) + \frac{c}{2} \sum_{j=1}^{n} \psi_{\underline{j}}(x) \right]^2$$

We need to expand the expression in order to find the elementary landscape decomposition. Due to space constraints we omit the intermediate steps and present the final expressions of the elementary components of f^2:

$$\left(f^2\right)^{(0)}(x) = \beta^2 + \frac{c^2}{4}n - \sum_{i=1}^{k} \frac{c|V_i| + 2\beta}{2^{|V_i|}} + \sum_{i,i'=1}^{k} \frac{1}{2^{|V_i \cup V_{i'}|}} \tag{35}$$

$$\begin{aligned}\left(f^2\right)^{(1)}(x) = {} & c\beta(n - 2ones(x)) - \sum_{i=1}^{k} \left(\frac{(c|V_i| + 2\beta)(-1)^{n_1^{(i)}}}{2^{|V_i|}} \mathcal{K}^{|V_i|}_{|V_i|-1,n_1^{(i)}} \right) \\ & + \sum_{i,i'=1}^{k} \left(\frac{(-1)^{n_1^{(i \vee i')}}}{2^{|V_i \cup V_{i'}|}} \mathcal{K}^{|V_i \cup V_{i'}|}_{|V_i \cup V_{i'}|-1,n_1^{(i \vee i')}} \right) \\ & - c \sum_{i=1}^{k} \frac{n - 2ones(x) - |V_i| + 2n_1^{(i)}}{2^{|V_i|}}\end{aligned} \tag{36}$$

$$\begin{aligned}\left(f^2\right)^{(2)}(x) = {} & \frac{c^2}{2}(-1)^{ones(x)} \mathcal{K}^n_{n-2,ones(x)} - \sum_{i=1}^{k} \left(\frac{(c|V_i| + 2\beta)(-1)^{n_1^{(i)}}}{2^{|V_i|}} \mathcal{K}^{|V_i|}_{|V_i|-2,n_1^{(i)}} \right) \\ & + \sum_{i,i'=1}^{k} \left(\frac{(-1)^{n_1^{(i \vee i')}}}{2^{|V_i \cup V_{i'}|}} \mathcal{K}^{|V_i \cup V_{i'}|}_{|V_i \cup V_{i'}|-2,n_1^{(i \vee i')}} \right) \\ & - c \sum_{i=1}^{k} \frac{(-1)^{n_1^{(i)}}}{2^{|V_i|}} \mathcal{K}^{|V_i|}_{|V_i|-1,n_1^{(i)}} \left(n - 2ones(x) - |V_i| + 2n_1^{(i)} \right)\end{aligned} \tag{37}$$

$$\left(f^2\right)^{(p)}(x) = -\sum_{i=1}^{k}\left(\frac{(c|V_i|+2\beta)(-1)^{n_1^{(i)}}}{2^{|V_i|}}\mathcal{K}^{|V_i|}_{|V_i|-p,n_1^{(i)}}\right)$$
$$+\sum_{i,i'=1}^{k}\left(\frac{(-1)^{n_1^{(i\vee i')}}}{2^{|V_i\cup V_{i'}|}}\mathcal{K}^{|V_i\cup V_{i'}|}_{|V_i\cup V_{i'}|-p,n_1^{(i\vee i')}}\right)$$
$$-c\sum_{i=1}^{k}\frac{(-1)^{n_1^{(i)}}}{2^{|V_i|}}\mathcal{K}^{|V_i|}_{|V_i|-p+1,n_1^{(i)}}\left(n-2ones(x)-|V_i|+2n_1^{(i)}\right) \quad (38)$$

where $\beta = k - cn/2$, $n_1^{(i\vee i')}$ are the number of ones in the positions x_j of the solution with $j \in V_i \cup V_{i'}$ and $p > 2$. The elementary components (36), (37) and (38) can be computed in $O(nk^2)$. Furthermore, we found an algorithm which computes all (not only one) the components in $O(nk^2)$.

5 Application of the Decomposition

In Section 4 we have derived closed-form formulas for each elementary component of f and f^2. Using this decompositions we can compute the average μ_1 and the standard deviation σ of the fitness distribution in the spheres and balls of arbitrary radius around a given solution x. Once we have the evaluation of the elementary components, the first and second order moments of f, μ_1 and μ_2, can be computed from Eqs. (32)-(34) and (35)-(38) in $O(n)$ for any ball or sphere around the solution using (15). The standard deviation can be computed from the two first moments using the equation $\sigma = \sqrt{\mu_2 - \mu_1^2}$.

How can we use this information? We propose here the following operator. Given a solution x compute the μ_1 and σ of the fitness distribution around the solution in all the spheres and balls up to a maximum radius r. We can do this in $O(nk^2)$, assuming that r is fixed. Using the averages and the standard deviations computed, we check if there is a high probability of finding a solution in a region around x that is better than the best so far solution. This check is based on the expression $\mu_1 + d \cdot \sigma - best$, where d is parameter and $best$ is the fitness value of the best so far solution. The higher the value of the previous expression, the higher the probability of finding a solution in the corresponding region that is better than the best solution. The previous expression is based on the idea that most of the samples of a distribution can be found around the average at a distance that is a few times the standard deviation. For example, at least 75% of the samples can be found in the interval $[\mu_1 - 2\sigma, \mu_1 + 2\sigma]$. In the case of the normal distribution, the percentage is 95%. In our operator, if $\mu_1 + d \cdot \sigma > best$, then it is likely that a solution better than the best found can be inside the considered region. If that happens, then a local search is performed in the region. This local search evaluates all the solutions in that region and replaces the current one by the best solution found. The pseudocode of the operator is in Algorithm 1.

We call this operator Guarded Local Search (GLS) because it applies the local search only in the case that there exists some evidence for the success. In addition, the local search is performed in the region in which most probably a

Algorithm 1 Pseudocode of the GLS operator

```
1: best = best so far solution;
2: bestRegion = none;
3: quality = −∞;
4: for r ∈ all the considered regions do
5:    (μ1,σ) = computeAvgStdDev(x,r);
6:    if μ1 + d · σ − best > quality then
7:       quality = μ1 + d · σ − best;
8:       bestRegion = r;
9:    end if
10: end for
11: y=x;
12: if quality > 0 then
13:    y = applyLocalSearchInRegion (x,bestRegion)
14: end if
15: return y
```

better solution would be found, thus minimizing the computation cost of a local search in a larger region. We expect our proposed operator to have an important intensification component. Thus, a population-based metaheuristic would be a good complement to increase the diversification of the combined algorithm. The operator can improve the quality of solutions of the algorithm it is included in, but it also will increase the runtime. However, this runtime should be quite lower than the one obtained if the local search would be applied at every step of the algorithm.

5.1 Experimental Study

As a proof of concept, we analyze the performance of the proposed operator in this section. For this experimental study we use a steady-state Genetic Algorithm (GA) with 10 individuals in the population, binary tournament selection, bit-flip mutation with probability $p = 0.01$ of flipping a bit, one-point crossover and elitist replacement. The stopping condition is to create 100 individuals (110 fitness evaluations). We compare three variants of the GA that differ in how the local search is applied. The first variant does not include any local search operator. In the second variant, denoted with GLSr, the GLS operator of Algorithm 1 is applied to the offspring after the mutation. The regions considered are all the spheres and balls up to radius r. The third variant, LSr, always applies the local search after the mutation in a ball of radius r.

For the experiments we selected six programs from the Siemens suite. The programs are `printtokens`, `printtokens2`, `schedule`, `schedule2`, `totinfo` and `replace`. They are available from the Software-artifact Infrastructure Repository [2]. Each program has a large number of available test suites, from which we select the first 100 tests covering different nodes. Thus, in our experiments $n = 100$. The constant tuning the oracle cost was set to $c = 1$. We used three values for the radius r: from 2 to 4. In the GLS the parameter d was set to $d = 2$.

Since we are dealing with stochastic algorithms we performed 30 independent executions and we show in Table 1 the average values obtained for the fitness of the best solution found and the execution time of the algorithms, respectively.

Table 1. Fitness of the best solution found and computation time (in seconds) of the algorithms (averages over 30 independent runs)

Alg.	printtokens		printtokens2		schedule		schedule2		totinfo		replace	
	Fit.	Secs.	Fit.	Secs.	Fit.	Secs.	Fit.	Secs.	Fit.	Secs.	Fit.	Secs.
GA	89.20	0.03	103.13	0.10	84.57	0.07	78.70	0.10	86.87	0.03	71.90	0.03
GLS2	105.17	37.93	119.63	69.73	101.60	21.10	93.60	52.63	102.30	39.07	88.13	37.30
LS2	113.27	10.67	129.00	20.73	111.07	3.80	103.10	3.17	110.00	3.03	97.67	5.53
GLS3	106.33	136.97	120.87	84.10	103.40	31.80	95.30	29.90	103.03	33.40	90.73	60.73
LS3	113.63	159.30	129.80	141.33	111.80	298.07	103.97	90.67	110.00	88.13	98.00	141.37
GLS4	105.27	390.03	121.47	363.53	103.40	237.17	96.37	212.70	104.33	206.50	91.13	368.97
LS4	114.00	3107.47	129.97	2943.03	112.00	2098.00	104.00	1875.67	110.00	1823.80	98.00	3602.47

We can observe in Table 1 that the ordering of the algorithms according to the solutions quality is LSr > GLSr > GA. This is the expected result, since LSr always applies a depth local search while GLSr applies the local search only in some favorable circumstances. An analysis of the evolution of the best fitness value reveals that this ordering is kept during the search process.

If we focus on the computation time required by the algorithms, we observe that GA is always the fastest algorithm. When $r \geq 3$, GLSr is faster than LSr. However, if $r = 2$ then LSr is faster than GLSr. This means that the complete exploration of a ball of radius $r = 2$ is faster than determining if a local search should be applied in the GLS operator. Although we show here the computation times, it should be noted that this depends on the implementation details and the machines used. For this reason the stopping condition is the number of evaluations. The great amount of time required to compute the elementary components is the main drawback of the GLS operator. However, this computation can be parallelized, as well as the application of the local search. In particular, Graphic Processing Units (GPUs) can be used to compute the elementary components in parallel.

6 Conclusion

We have applied landscape theory to find the elementary landscape decomposition of the Test Suite Minimization problem in regression testing. We have also decomposed the squared objective function. Using the closed-form formulas of the decomposition we can compute the average and the standard deviation of the fitness values around a given solution x in an efficient way. With these tools we proposed an operator to improve the quality of the solutions. This operator applies a local search around the solution only if the probability of finding a best solution is high. The results of an experimental study confirms that the operator improves the solutions requiring a moderate amount of computation. A blind local search outperforms the results of our proposed operator but requires a large amount of computation as the size of the explored region increases.

The future work should focus on new applications of the theory but also on new theoretical implications of the elementary landscape decomposition, such as determining the difficulty of a problem instance by observing its elementary

components or predicting the behaviour of a search algorithm when applied to a problem.

Acknowledgements

We thank the anonymous reviewers for their interesting and fruitful comments. This research has been partially funded by the Spanish Ministry of Science and Innovation and FEDER under contract TIN2008-06491-C04-01 (the M* project) and the Andalusian Government under contract P07-TIC-03044 (DIRICOM project).

References

1. Chicano, F., Whitley, L.D., Alba, E.: A methodology to find the elementary landscape decomposition of combinatorial optimization problems. Evolutionary Computation In press (doi: 10.1162/EVCO_a_00039)
2. Do, H., Elbaum, S., Rothermel, G.: Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact. Empirical Softw. Engg. 10, 405–435 (October 2005), http://portal.acm.org/citation.cfm?id=1089922.1089928
3. Feinsilver, P., Kocik, J.: Krawtchouk polynomials and krawtchouk matrices. In: Baeza-Yates, R., Glaz, J., Gzyl, H., Hüsler, J., Palacios, J. (eds.) Recent Advances in Applied Probability, pp. 115–141. Springer US (2005)
4. Langdon, W.B.: Elementary bit string mutation landscapes. In: Beyer, H.G., Langdon, W. (eds.) Foundations of Genetic Algorithms. pp. 25–41. ACM, Schwarzenberg, Austria (5-9 Jan 2011)
5. Lu, G., Bahsoon, R., Yao, X.: Applying elementary landscape analysis to search-based software engineering. In: Proceedings of the 2nd International Symposium on Search Based Software Engineering (2010)
6. Rana, S., Heckendorn, R.B., Whitley, D.: A tractable walsh analysis of SAT and its implications for genetic algorithms. In: Proceedings of AAAI. pp. 392–397. Menlo Park, CA, USA (1998)
7. Reidys, C.M., Stadler, P.F.: Combinatorial landscapes. SIAM Review 44(1), 3–54 (2002)
8. Stadler, P.F.: Toward a theory of landscapes. In: López-Peña, R., Capovilla, R., García-Pelayo, R., H.Waelbroeck, Zertruche, F. (eds.) Complex Systems and Binary Networks. pp. 77–163. Springer-Verlag (1995)
9. Sutton, A.M., Howe, A.E., Whitley, L.D.: Directed plateau search for MAX-k-SAT. In: Proceedings of SoCS. Atlanta, GA, USA (July 2010)
10. Sutton, A.M., Whitley, L.D., Howe, A.E.: Computing the moments of k-bounded pseudo-boolean functions over Hamming spheres of arbitrary radius in polynomial time. Theoretical Computer Science In press (doi: 10.1016/j.tcs.2011.02.006)
11. Sutton, A.M., Whitley, L.D., Howe, A.E.: A polynomial time computation of the exact correlation structure of k-satisfiability landscapes. In: Proceedings of GECCO. pp. 365–372. ACM, New York, NY, USA (2009)
12. Whitley, D., Sutton, A.M., Howe, A.E.: Understanding elementary landscapes. In: Proceedings of GECCO. pp. 585–592. ACM, New York, NY, USA (2008)
13. Yoo, S., Harman, M.: Pareto efficient multi-objective test case selection. In: Proceedings of the 2007 International Symposium on Software Testing and Analysis (ISSTA '07). pp. 140–150. ACM, London, England (9-12 July 2007)
14. Yoo, S., Harman, M.: Regression testing minimisation, selection and prioritisation: A survey. Journal of Software Testing, Verification and Reliability (2010)