Elementary Bit String Mutation Landscapes

W. B. Langdon
CREST centre, Department of Computer Science,
University College, London, Gower Street, London, WC1E 6BT, UK
w.langdon@cs.ucl.ac.uk

ABSTRACT

Genetic Programming parity with only XOR is not elementary. GP parity can be represented as the sum of $k/2+1$ elementary landscapes. Statistics, including fitness distance correlation (FDC), of Parity's fitness landscape are calculated. Using Walsh analysis the eigen values and eigenvectors of the Laplacian of the two bit, three bit, n-bit and mutation only Genetic Algorithm fitness landscapes are given. Indeed all elementary bit string landscapes are related to the discrete Fourier functions. However most are rough ($\lambda/d \approx 1$). Also in many cases fitness autocorrelation falls rapidly with distance. GA runs support eigenvalue/graph degree (λ/d) as a measure of the ruggedness of elementary landscapes for predicting problem difficulty. The elementary needle in a haystack (NIH) landscape is described.

Categories and Subject Descriptors

F.2.m [**Analysis of Algorithms and problem Complexity**]: Miscellaneous; G.2.2 [**Graph Theory**]; G.1.6 [**Optimisation**]: Stochastic programming; G.3 [**Probability and Statistics**]: Probabilistic algorithms; I.2.8 [**Artificial Intelligence**]: Problem Solving, Search

General Terms

Theory

Keywords

genetic algorithms, genetic programming, search, optimisation, graph theory, Laplacian, Hamming cube, Walsh transform, fitness distance correlation, elementary fitness autocorrelation

1. FITNESS LANDSCAPES

An elementary landscape is a special case of a fitness landscape. There are some important combinatorially hard problems, e.g. k-satisfiability and maxsat [21], which have elementary landscapes and where these properties have been used to devise improved search techniques. This is the first time genetic programming has been analysed in terms of elementary landscapes. Our motivation is to explore the technique, with a view to further analyse and hopefully improve GP. However to do this we recast GP as a bit string genetic algorithm (GA) and report many problem independent properties of elementary landscapes.

Firstly we recap fitness landscapes then some well known properties of elementary landscapes will be described in the next section. Section 3 describes our simplified version of the genetic programming parity problem. In Section 4 we approximate tree GP parity by a GA where mutation flips exactly two bits. Section 5 gives many properties of the fitness landscapes created by this double bit flip mutation, particularly those relating to GP parity. In Section 6 some of these results are generalised to three and n-bit flip mutation as well as to normal mutation only GAs. Section 7 describes some GP experiments on parity and on two elementary landscapes.

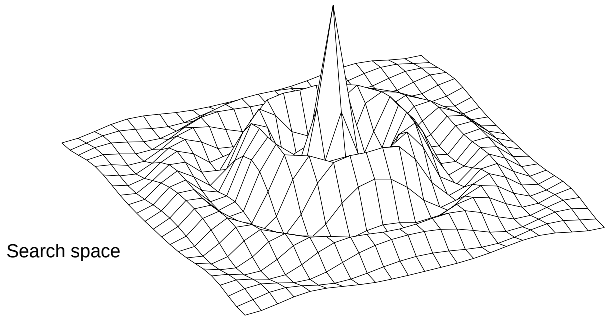

Figure 1: A fitness landscape where internal nodes have four neighbours. Fitness is plotted vertically.

Fitness landscapes (see Figure 1) have often been used to try and explain how optimisation techniques, particularly evolutionary algorithms [1] such as genetic programming [13] work. An optimisation problem can viewed as a search problem where all possible solutions to the problem are nodes in the search space and each has a value. In genetic algorithms [15] this is called a fitness value (more generally an objective value). Optimisation is viewed as sampling from this space with the aim of to finding better points (or even the best point) in the space.

Except for Monte Carlo methods, optimisation techniques use information gathered from previous samples to decide where in the search space to sample next. The goal being to minimise the number of samples that are needed before an acceptable solution is found. In general evolutionary algorithms, and several other optimisation techniques, only use the current search point (or the current population of search points) to guide the choice of where to look next. They do not use previously gained knowledge. Different algorithms can have radically different ways of moving from one point in the search space to the next. A search neighbourhood is the set of points that a specific algorithm can reach in one step from the current search point. A fitness landscape can be thought of as a graph where neighbours are linked by a single edge if and only if our search algorithm can move between the two nodes in the graph. While the height of the node is given by its fitness.

Typically in evolutionary algorithms, the edges in the graph are undirected, because if one node can be reached from another then the reverse move is also possible. Notice that the difficulty of a problem depends not only on how fitness values are decided but also on the way the search algorithm moves across the search space. I.e. problem difficulty also depends on the fitness landscape the search algorithm imposes on the underlying problem [11, Chp. 2].

In genetic algorithms a common fitness landscape is to encode candidate solutions as strings of l bits. Each of the 2^l bit strings is allocated a fitness value. This gives a search space of 2^l candidate solutions. If mutation is restricted to flipping a single bit then each of the candidate solutions is connected in the fitness landscape to l other candidate solutions. This is known as the Hamming neighbourhood or hypercube graph [22]. The fitness landscape metaphor can be extended to population approaches (such as Particle Swarm Optimisation [12]) by allowing multiple sample points (one for each member of the population) in the landscape.

Fitness landscapes are a useful metaphor with simple mutation which has a well defined neighbourhood. However with the canonical mutation only genetic algorithm the analogy starts to fail. Since GA mutation is defined as a probability of each bit flipping, multiple bits can be changed. There is a finite, albeit exponentially small, probability of any string being converted into any other. Thus, in principle, the whole fitness landscape becomes a single fully connected clique.

Fitness landscapes can be extended to allow multi-parent search operations (e.g. crossover). However then the neighbourhood of each member of the population depends upon the location of everyone else in the population and the idea of fixed predetermined links in the graph fails. (Nevertheless Stadler and Wagner extend the idea of fitness landscapes to include crossover by using P-structures [19].)

2. ELEMENTARY LANDSCAPES

2.1 Wave Equation

The following is based on Whitley's (e.g. [25] and [27]) definitions. (See also Grover [6] and Stadler [18, p3].)

In an elementary landscape the fitness function f and the search space neighbourhood graph are related by a wave equation. The search space neighbourhood graph can be represented by a matrix Δ (see Sections 2.2 and 5.3). Only in an elementary landscape do the search space and the fitness function obey

$$\Delta f = \lambda(f - \overline{f})$$

Where $\overline{f}$ is the mean fitness across the whole search space. That is, zero-mean fitness $f - \overline{f}$ is an eigenvector of Δ with eigenvalue λ. Notice a given search space (e.g. that created by one point mutation acting on a bit string chromosome) will have multiple eigenvectors. Each eigenvector will be the fitness function (up to an additive constant) for a different elementary landscape.

2.2 Average Fitness Change

For simplicity we will deal with regular landscapes. I.e. landscapes where every location has the same number of search neighbours d. In general the mean change in fitness caused by one genetic change depends on the current position in the search space. Treating the mean change as a vector of the same size as the search space gives it as:

$$\frac{1}{d}(Af - f)$$

If we treat the search space as a graph, then d is the degree of each node in the graph. (I.e. the number of links from the node). A is the adjacency matrix. For our purposes, it is a sparse real square matrix where every element corresponding to a link in the graph has the value 1 and all the others are zero.

$$\frac{1}{d}(Af - f) = \frac{1}{d}(A - dI)f = -\frac{1}{d}\Delta f$$

Where Δ is the Laplacian and is defined to be $dI - A$. (I is the identity matrix.)

2.3 Average Neighbourhood Fitness in an Elementary Landscape

The mean fitness of a neighbourhood, $N(x)$, will also depend on the current position in the search landscape x. It is given by the fitness of the current position, $f(x)$, plus the average change in fitness (calculated in the previous section).

$$\begin{aligned}
\text{avg}_{y\in N(x)}\{f(y)\} &= \frac{1}{d}\sum_{y\in N(x)} f(y) \\
&= f(x) + \frac{1}{d}\sum_{y\in N(x)} f(y) - f(x) \\
&= f(x) - \frac{1}{d}\Delta f(x) \\
&= f(x) - \frac{1}{d}\lambda(f(x) - \overline{f}) \\
&= f(x) + \frac{\lambda}{d}(\overline{f} - f(x))
\end{aligned}$$

If, for convenience we set $\overline{f}$ to zero, in an elementary landscape the mean fitness of the neighbours of x is:

$$\text{avg}_{y\in N(x)}\{f(y)\} = f(x) - \frac{\lambda}{d}f(x) = (1 - \frac{\lambda}{d})f(x) \qquad (1)$$

Thus, if $\lambda < d$, the mean of the neighbourhood is always closer to the overall average $\overline{f}$ than the centre of the neighbourhood $f(x)$ is. Figure 2 shows a local optimum. By Equation 1, the average of the neighbourhood must lie between $\overline{f}$ and $f(x)$. At a local optimum, by definition, $f(x)$ must be above the fitness of all its neighbours and hence must be above their average fitness. Therefore $f(x)$ must be above $\overline{f}$. Another way of saying this is: in elementary landscapes (with $\lambda < d$) there are no local optima with below average fitness. In special cases, e.g. k-Satisfiability and

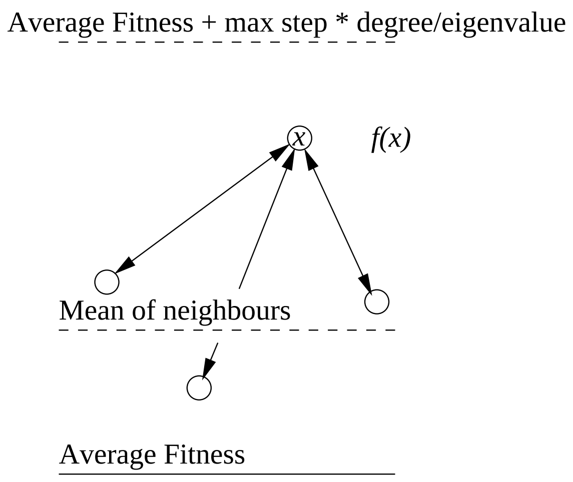

Figure 2: By definition at a hill top x the fitness of its neighbours is lower than that of the peak $f(x)$. By Equation 1, provided $\lambda < d$, the average fitness of the neighbours of x is closer to the average of the whole landscape $\overline{f}$ than $f(x)$ is. Therefore $f(x)$ must lie above $\overline{f}$ [6, Thm. 6]. Dimova *et al.* give an upper bound $f \leq \overline{f} + \epsilon_{\max} d/\lambda$ [5, Crly. 4].

MAXSAT [21], there are tighter bounds on the fitness of local optima and even on the widths of plateaus. (See also [20].)

2.4 Bound on Maximum Fitness

Dimova *et al.* [5, Crly. 4] prove a global upper bound $f \leq \overline{f} + \epsilon_{\max} d/\lambda$ for the whole search space. Where $\epsilon_{\max}$ is the largest change in fitness for any single step in the landscape [5, 23]. (Remember d is the number of possible moves and λ is Δ's eigenvalue.) For onemax: $\overline{f}$=$l/2$, d=l and the eigenvalue λ=2. Since every move either increments or decrements fitness $\epsilon_{\max}$=1. Thus in the case of onemax, $f_{\max} = \overline{f} + \epsilon_{\max} d/\lambda$ holds exactly at the global maximum $f_{\max} = l$ and f is strictly less than $\overline{f} + \epsilon_{\max} d/\lambda$ elsewhere.

As a second example, take the needle in a haystack (which will be fully described in Section 5.4). The largest step $\epsilon_{\max}$ is 1, the mean fitness $\overline{f}$ is 2^{-n}, node degree d is 2^n and the eigenvalue λ=2^n. Thus $f_{\max} = \overline{f} + \epsilon_{\max} d/\lambda = 1 + 2^{-n}$, which is very close to the global maximum (1).

As a third example, take the concatenated four bit trap like elementary landscape described in Section 7.3. The average fitness $\overline{f}$=2×4=8 and $f_{\max}$=4×4=16. (The full 16-bit fitness function is given by adding together four functions like that given in Figure 5). Since mutation flips exactly two bits, the number of possible moves is d=C_2^{16}=16 × 15/2=120. The eigenvalue λ=84. The maximum change in fitness comes when mutation flips two bits in separate traps so each causes the maximum change (3). Adding these together we get $\epsilon_{\max}$=6. Thus $\overline{f} + \epsilon_{\max} d/\lambda = 8 + 6 \times 120/84 = 16+4/7$. Again, as expected, $f_{\max} < \overline{f} + \epsilon_{\max} d/\lambda$ holds everywhere including the global maximum. Notice that the bound can be calculated directly. I.e. without sampling the search space at all.

In the case of 4×Trap-4 we know fitness values have integer values, so if we find a point with fitness 16, Dimova *et al.*'s bound tells us it is a global optimum. (But not that it is the unique global optimum.) More generally it places a limit on how much a result can be improved. E.g. if we have found a 4×Trap-4 point with value 15, we know that further search can only improve by at most 1. This could be used as another way of deciding when to stop searching.

3. GENETIC PROGRAMMING PARITY

The classic definition of the GP parity problems was given by Koza [8]. He defined the representation for order-k parity as binary trees whose external leafs are the functions inputs (drawn from $D_0 \cdots D_{k-1}$) and whose internal nodes are drawn from four binary Boolean functions. Initially the first simplification is to use just one Boolean function rather than four. We use either equals (EQ) or exclusive-or (XOR) depending if we are dealing with even or odd parity.

Koza defines the fitness of each tree by testing it on all 2^k possible input patterns and counting the number of times it returns the same answer as parity would. (Even-k-parity is true if the number of inputs which are true is even.) Thus Koza's fitness lies in the range $0 \ldots 2^k$ and a solution to the parity problems has fitness 2^k.

Elsewhere [10, pages 421–423] we have used the symmetries of EQ and XOR to show that with a function set composed only of either EQ or only of XOR, trees will have fitness of either 2^{k-1} or 2^k. These properties also carry over from tree based GP to Cartesian GP and have also been exploited by Yu and Miller [29]. Further a tree is only a solution to parity if it contains an odd number of each type of leaf. (To simplify the text, and since odd parity behaves similarly, from now on we will discuss only even parity.) Since solutions to k-parity have one leaf of each of the k types plus redundant pairs of the same leaf type, they always have $k + 2n$ leafs (for $n = 0, 1, 2, \ldots$). So their total size is $2k + 4n - 1$. Notice neither the shape of the tree nor the order of its leafs matters. All binary trees formed by re-arrangements of the leafs and internal nodes of the tree have identical fitness. The group properties of EQ also mean any pairs of leafs of the same type can be removed. E.g., using = to represent EQ, $(D_1 = (D_2 = D_2))$ is identical to the input D_1 on its own. So if there are an even number any type of leaf (e.g. D_2) then it is as if the input was not connected. A tree missing one or more inputs scores exactly half the maximum score on parity.

We extend our previous analysis of the parity problem [10], which described its fitness distribution, to consider parity's fitness landscape.

4. THE PARITY PROBLEM

4.1 The Mutation Operator

To form a landscape, in addition to the representation and its associated fitness function we need at least one search operator to establish which representations are adjacent to each other. Koza's [8] subtree crossover is a bit complicated to start with so we shall instead start with a mutation operator.

The mutation operator changes exactly one thing in the tree. Since the internal nodes are always EQ, the only possible change is to convert one leaf from one input (D_i) to another (D_j with $j \neq i$). This does not change the size of the tree. In principle more complicated tree size changing mutation operators or indeed crossover operators might also be considered.

4.2 Mutation's Impact on Fitness

The effect of mutation is either to make no difference to fitness (i.e. remain at 2^{k-1}) or to increase it from 2^{k-1} to 2^k or reduce it from 2^k to 2^{k-1}. A solution to parity has an odd number of all k types of leafs. Thus replacing any leaf with another of a different type will always mean it is no longer a solution. So it is impossible to mutate a tree of fitness 2^k into another tree of fitness 2^k.

Now the chance of each of these three things happening depends only on the number of each type of leaf. It does not depend upon the order of the leafs or their placement on the tree. Since these permutations make no difference to either fitness or to any of the changes in fitness we ignore them and initially replace trees with k integer counts d_i. Obviously there is a constraint that all of the tree's leafs must be present. So for a tree of size $2k + 4n - 1$, $\sum_{i=0}^{k-1} d_i$ must be $k + 2n$.

A tree's fitness is only 2^k if the number of leafs of each type (d_i) is odd for all k types. We can further simplify the representation by replacing the d_is with e_is which are 1 if their corresponding d_i is even and 0 if it is odd. Similarly we can simplify Koza's fitness function so that 2^{k-1} is replaced by 0 and 2^k by 1. Under the new scheme fitness = 1 if and only if all e_i are 0 and fitness = 0 otherwise.

Mutation cannot change the total number of leafs, therefore having chosen an initial tree, $\sum d_i$ is fixed. This is a constraint on d_i and so we only need to specify $k - 1$ values for the d_i. (The remaining one can be inferred from the need to supply all the tree's leafs.) In fact, the oddness or evenness of $(k - 1)$ d_i is enough for the odd or evenness of the remaining one to be inferred. Another way of looking at this is to say: although for convenience we have k e_i, mutation can only reach half of their 2^k possible values. The unreachable half of the 2^k values correspond to trees of the wrong size for any of them to be parity solutions and so have zero fitness. We will assume the tree size is $2k + 4n - 1$, which allows one solution in a space of 2^{k-1}.

Mutating a leaf from type i to type j means decrementing d_i and incrementing d_j. Provided $d_i > 0$, this is equivalent to inverting e_i and e_j. I.e. mutation flips exactly two of the k bits. The next section will describe how we ensure $d_i > 0$ so that all mutation are always possible and further they are all equally likely.

4.3 Large Trees Simplify Mutation Analysis

Now it will greatly simplify things later if we assume the type of input we are about to mutate is chosen uniformly at random from the k possible types and that the type of the replacement leaf is chosen uniformly at random from the $k - 1$ remaining types. Obviously this requires the tree to have at least one leaf of each of the k types.

To further simplify the analysis we shall assume that the trees are much bigger than the minimum size ($2k-1$). So big in fact, that we can assume that the tree has more than one leaf of each type. Further we assume the tree has sufficiently large number of leafs that undirected random drift from an initial random starting point in the search landscape will never cause the number of leafs of any of the k types to fall to one. I.e. random drift will not approach the edge of the k-dimensional simplex. If evolution lasts for G generations then drift will change the number of leafs of a given type by about $\sqrt{G}$. So assuming the tree is also bigger than a constant multiple of $(2k - 1)\sqrt{G})$ will ensure the chance of any of the k types of leaf approaching extinction is negligible. So mutation is always free to choose any pair of leaf types. This ensures it is always symmetric.

5. THE PARITY FITNESS LANDSCAPE

Treat the landscape as a graph where the nodes are k length bit vectors. As in Section 4.2, if bit i is 1 this indicates the tree has an even number of D_i leafs. Nodes in the graph are directly connected (i.e. the corresponding trees are neighbours) if mutating one node gives the other in exactly one step. We deal with the 2^{k-1} nodes that are indirectly connected to the solution node.

Neighbours in the graph have exactly two bits different. Thus the graph consists of 2^{k-1} nodes each connected by $\frac{1}{2}k(k - 1)$ symmetric links and the probability of moving along each link is the same (i.e. $\frac{2}{k(k-1)}$). Only the single node with none of the k elements are even has fitness 1. All other nodes have zero fitness. Average fitness is $\overline{f} = 2^{1-k}$. The variance of fitness is $\frac{1}{2^{k-1}} \sum f_i^2 - \overline{f}^2 = \frac{1}{2^{k-1}} - \overline{f}^2 = 2^{1-k} - 2^{2-2k}$. So the standard deviation $\sigma_f = \sqrt{2^{1-k} - 2^{2-2k}} \approx 2^{-(k-1)/2}$.

5.1 Fitness Distance Correlation

Jones and Forrest state that the fitness distance correlation based on random sampling of a needle in a haystack fitness function will be near zero [7, page 186]. We confirm this by giving values based on analysing the whole space and thus avoiding noise introduced by random sampling.

Jones and Forrest [7, page 185] define the fitness distance correlation as $r = \mathrm{Cov}(f, d)/\sigma_f \sigma_d$. Where the covariance between fitness and distance (d) to the global optimum is: $\mathrm{Cov}(f, d) = \frac{1}{2^{k-1}} \sum_{i=0}^{2^{k-1}-1} (f_i - \overline{f})(d_i - \overline{d})$ and σ_f is the standard deviation of f (calculated in the previous section) and similarly σ_d is the standard deviation of the number of two bit flips to the origin (i.e. distance to the global optimum).

$$
\begin{aligned}
&\mathrm{Cov}(f, d)\\
&= \frac{1}{2^{k-1}} \sum_{i=0}^{2^{k-1}-1} (f_i - \overline{f})(d_i - \overline{d})\\
&= \frac{1}{2^{k-1}} \left((1 - \overline{f})(d_0 - \overline{d}) + \sum_{i=1}^{2^{k-1}-1} -\overline{f}(d_i - \overline{d}) \right)\\
&= \frac{1}{2^{k-1}} \left((1 - \overline{f})(d_0 - \overline{d}) - \overline{f} \sum_{i=1}^{2^{k-1}-1} d_i - \overline{d} \right)\\
&= \frac{1}{2^{k-1}} \left((1 - \overline{f})(-\overline{d}) + \overline{f}(d_0 - \overline{d}) - \overline{f} \sum_{i=0}^{2^{k-1}-1} d_i - \overline{d} \right)\\
&= \frac{1}{2^{k-1}} \left((1 - \overline{f})(-\overline{d}) - \overline{f}\,\overline{d} \right)\\
&= \frac{-\overline{d}}{2^{k-1}} \left((1 - \overline{f}) + \overline{f} \right)\\
&= \frac{-\overline{d}}{2^{k-1}}
\end{aligned}
$$

The distance from the optimum is $\approx bc/2$. Where bc is the number of 1s (the bit count). The number of points in

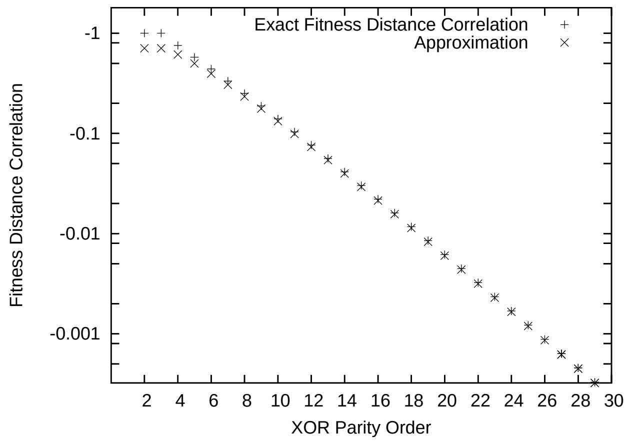

Figure 3: Fitness Distance Correlation for Needle in a Haystack with double flip mutation. Note log scale.

the search space with identical bit count is C_{bc}^{k-1}. C_{bc}^{k-1} are coefficients of the binomial distribution. The mean of the binomial distribution is np and the variance is $np(1-p)$. Here $p = 1/2$ and $n = k-1$. So the mean distance divided by the standard deviation of the distance $\overline{d}/\sigma_d \approx n^{\frac{1}{2}} = \sqrt{k-1}$.

$$\begin{aligned} r = \frac{\mathrm{Cov}(f,d)}{\sigma_f \sigma_d} &\approx \frac{-\overline{d}}{2^{k-1}} \frac{1}{2^{-(k-1)/2}\ \sigma_d} \\ &= \frac{-1}{2^{(k-1)/2}} \frac{\overline{d}}{\sigma_d} \\ r &\approx \frac{-\sqrt{k-1}}{2^{(k-1)/2}} \end{aligned}$$

Figure 3 plots this approximation and the exact fitness distance correlation. It shows even for quite modest order, the actual correlation coefficient converges to this large order approximation.

5.2 Walsh Analysis of Fitness Autocorrelation

Jones [7] defined fitness distance correlation in terms of distance from the global optimum. This has a number of problems. Usually one needs to know the location of the best solution in the search space and it is assumed that there is only one solution with the best fitness value. Fitness autocorrelation, for example along a random walk, addresses these problems and can be used as a measure of landscape smoothness and indicator of how easy a problem might be to solve. This section provides an informal argument that fitness correlation between neighbours falls rapidly with distance. It could be argued that in the case of a needle in a haystack landscape, such as parity, the chance of encountering the needle is sufficiently remote that fitness along a random walk will always be zero in practise. Nevertheless the following arguments can be applied to any landscape where the fitness landscapes created by the Walsh functions are elementary landscapes. This includes 1-bit, 2-bit, 3-bit and n-bit flip mutation as well as GA bit string mutation (see Sections 5.6 and 6).

We shall see in Section 5.10 that GP Parity can be represented as a linear sum of elementary landscapes. Dimova *et al.* [4] proved that fitness autocorrelation for a random walk on an elementary landscapes falls exponentially. Since the autocorrelation of all the components of parity fall monotonically, the correlation of parity itself must also fall monotonically with distance. (With increasing distance the actual value will be dominated by the slowest changing exponential term.)

Using Walsh analysis, any discrete fitness landscape can be represented as a sum of Walsh coefficients. If (as is known in many cases) the Walsh basis functions are elementary landscapes and fitness correlation falls rapidly in one step it will continue to fall towards zero for all larger distances.

For simplicity let the mean fitness be zero. Then fitness auto-correlation is essentially given by:

$$\sum_i \sum_{id} f_i f_{id}$$

where $\sum_i$ is the sum over the whole search space and $\sum_{id}$ is the sum over all neighbours of i which are d steps from i. For illustration assume $f = w + v$ where w and v are two Walsh basis elementary landscapes.

$$\begin{aligned} \sum_i \sum_{id} f_i f_{id} &= \sum_i \sum_{id} (w_i + v_i)(w_{id} + v_{id}) \\ &= \sum_i \sum_{id} w_i w_{id} + v_i w_{id} + w_i v_{id} + v_i v_{id} \end{aligned}$$

Now the first and last terms are simply the auto-correlation of w and v which we know fall exponentially.

$$\begin{aligned} \sum_i \sum_{id} v_i w_{id} &= \sum_i \sum_{id} v_i (w_i - (w_i - w_{id})) \\ &= \sum_i D v_i w_i - \sum_i \sum_{id} v_i (w_i - w_{id}) \end{aligned}$$

where D is the number of neighbours separated by d. Now the first term is zero, since w and v are orthogonal Walsh basis functions.

$$\sum_i v_i \sum_{id} (w_i - w_{id}) \le v_{\max} \sum_i \sum_{id} (w_i - w_{id})$$

When the separation distance becomes large wrt w's order $\sum_i \sum_{id} w_i - w_{id}$ will become small. Hence the autocorrelation between the sum of two Walsh functions falls rapidly with distance. This argument can be generalised to summing more than two elementary landscapes and holds for any fitness landscape where the Walsh basis functions are elementary landscapes.

Section 6 will show with many mutation operations, the Walsh basis functions are elementary landscapes. As well as parity, well known examples include max-3-sat [25]. Fitness distance correlation will fall rapidly with distance for all of them.

5.3 Laplacian of the Parity Landscape Graph

Form the Laplacian Δ matrix as a real square matrix whose rows and columns correspond to the 2^{k-1} nodes in the parity landscape. Every element is zero except the off diagonal terms corresponding to an edge in the graph and the diagonal. Since the graph's edges are bidirectional and each have the same probability, the graph is symmetric and we can make all the non-zero off-diagonal terms be -1. The diagonal elements are the number of edges connected to the corresponding node in the graph. This is $\frac{1}{2}k(k-1)$ for all nodes. I.e. every node in the graph has the same degree. ($d = \frac{1}{2}k(k-1)$.) Thus every row (and every column) sums to zero.

$\Delta = \frac{1}{2}k(k-1)I - A$ where A is the graph's adjacency matrix.

$$\Delta_2 = \begin{array}{c|cc} & 00 & 11 \\ \hline 00 & 1 & -1 \\ 11 & -1 & 1 \end{array}$$

$$\Delta_3 = \begin{array}{c|cccc} & 000 & 101 & 110 & 011 \\ \hline 000 & 3 & -1 & -1 & -1 \\ 101 & -1 & 3 & -1 & -1 \\ 110 & -1 & -1 & 3 & -1 \\ 011 & -1 & -1 & -1 & 3 \end{array}$$

$$\Delta_4 = \begin{array}{c|cccccccc} & 0000 & 1001 & 1010 & 0011 & 1100 & 0101 & 0110 & 1111 \\ \hline 0000 & 6 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \\ 1001 & -1 & 6 & -1 & -1 & -1 & -1 & 0 & -1 \\ 1010 & -1 & -1 & 6 & -1 & -1 & 0 & -1 & -1 \\ 0011 & -1 & -1 & -1 & 6 & 0 & -1 & -1 & -1 \\ 1100 & -1 & -1 & -1 & 0 & 6 & -1 & -1 & -1 \\ 0101 & -1 & -1 & 0 & -1 & -1 & 6 & -1 & -1 \\ 0110 & -1 & 0 & -1 & -1 & -1 & -1 & 6 & -1 \\ 1111 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & 6 \end{array}$$

$$\Delta_5 =$$
$$\begin{array}{c|cccccccccccccccc} 00000 & 10 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & 0 & -1 & 0 & 0 & 0 \\ 10001 & -1 & 10 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & 0 & -1 & 0 & -1 & 0 & 0 \\ 10010 & -1 & -1 & 10 & -1 & -1 & 0 & -1 & -1 & -1 & 0 & -1 & -1 & 0 & 0 & -1 & 0 \\ 00011 & -1 & -1 & -1 & 10 & 0 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & -1 \\ 10100 & -1 & -1 & -1 & 0 & 10 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & -1 & -1 & -1 & 0 \\ 00101 & -1 & -1 & 0 & -1 & -1 & 10 & -1 & -1 & 0 & -1 & 0 & 0 & -1 & -1 & 0 & -1 \\ 00110 & -1 & 0 & -1 & -1 & -1 & -1 & 10 & -1 & 0 & 0 & -1 & 0 & -1 & 0 & -1 & -1 \\ 10111 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & 10 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & -1 \\ 11000 & -1 & -1 & -1 & 0 & -1 & 0 & 0 & 0 & 10 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \\ 01001 & -1 & -1 & 0 & -1 & 0 & -1 & 0 & 0 & -1 & 10 & -1 & -1 & -1 & -1 & 0 & -1 \\ 01010 & -1 & 0 & -1 & -1 & 0 & 0 & -1 & 0 & -1 & -1 & 10 & -1 & -1 & 0 & -1 & -1 \\ 11011 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & 10 & 0 & -1 & -1 & -1 \\ 01100 & -1 & 0 & 0 & 0 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & 0 & 10 & -1 & -1 & -1 \\ 11101 & 0 & -1 & 0 & 0 & -1 & -1 & 0 & -1 & -1 & -1 & 0 & -1 & -1 & 10 & -1 & -1 \\ 11110 & 0 & 0 & -1 & 0 & -1 & 0 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & 10 & -1 \\ 01111 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & 10 \end{array}$$

For a landscape to be elementary its fitness function must have a special relationship with its move operator; it must obey the "wave equation" $\Delta f = \lambda(f - \overline{f})$ (see Section 2.1). I.e. treat the fitness function as a vector whose elements are the fitnesses of the corresponding nodes in landscape graph. Call this f. If the landscape is to be elementary f (up to an additive constant) must be an eigenvector of Δ with corresponding eigenvalue λ. For parity, f is a vector with 2^{k-1} elements all of which are zero except the first, which has the value 1. Thus Δf is simply the first column of Δ. The first element of the first column of Δ is $\frac{1}{2}k(k-1)$ and there are $\frac{1}{2}k(k-1)$ other elements whose values are -1. Thus the first column of Δ is not a simple scalar multiple of the fitness vector f and so f is not an eigenvector of Δ. Therefore the parity landscape is not elementary.

Section 5.5 and those following it will show not only is parity's landscape not elementary but also it cannot be decomposed into a small number of elementary landscapes.

5.4 The Elementary Needle in Haystack

We use the analysis from the previous section to construct another needle in a haystack (NIH) problem which is elementary. Without loss of generality we can keep the solution at all zeros and give it fitness one (all other points have zero fitness). Thus the NIH has the same fitness function as parity. There are 2^l points in the search space, so $\overline{f} = 2^{-l}$. For an NIH landscape to be elementary it must still obey the "wave equation" $\Delta f = \lambda(f - \overline{f})$ and Δf is again the first column of Δ. An elementary NIH Laplacian is:

$$\Delta_{\text{NIH}} = \begin{array}{ccccc} 2^l\text{-}1 & -1 & -1 & \dots & -1 \\ -1 & 2^l\text{-}1 & -1 & \dots & -1 \\ -1 & -1 & 2^l\text{-}1 & \dots & -1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & -1 & \dots & 2^l\text{-}1 \end{array}$$

which gives $\lambda = 2^l$. That is we can construct an elementary needle in a haystack landscape with a mutation operator which can move to any point in the search landscape in one step and all such points are equally likely. Next we show that Δ_{NIH} is the only elementary needle in a haystack landscape Laplacian.

Suppose the first row of an elementary needle Laplacian is $(a, b, c, d, \dots, z)$. Since the Laplacian is symmetric its first column is also $(a, b, c, d, \dots, z)$. a is the degree of the solution node. Since the solution must be accessible $a > 0$. Now $f = (1, 0, 0, 0, \dots, 0)$. So Δf is $(a, b, c, d, \dots, z)$ but since Δ represents an elementary NIH landscape $\Delta f = \lambda(f - 2^{-n})$. Thus $a = \lambda(1 - 2^{-n})$, so $\lambda > 0$. Also $b = -\lambda 2^{-n}$. Since the off diagonal terms of Δ are either 0 or -1, b = -1 and so λ must be 2^n. The other elements of the first row of Δ, $c, d, \dots, z$, must also be -1. Either the wave equation or the fact that Δ's rows sum to zero gives us $a = 2^n - 1$. Notice b = -1, c = -1, d = -1, …, z = -1 in any elementary needle in a haystack landscape. That is, in any elementary NIH landscape every point in the search space can reach the global optimum in one move and vice versa.

The last part of the argument, is to say that in a fair needle in a haystack problem, the chance of finding the optimum should be no more than the chance of finding any other point in the search space. Thus the number of paths into that point should not be less than the number into the optimum. We have shown the optimum's degree is $2^n - 1$ and so every row must contain $2^n - 1$ "-1" elements. Since each row has only 2^n elements, every off diagonal element must be -1. I.e. Δ_{NIH} represents the only elementary needle in a haystack landscape.

5.5 Recursive Construction of Parity's Landscape

We use the notation

$$\left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right)$$

to indicate a square $n \times n$ matrix that is composed of four square $n/2 \times n/2$ matrices A, B, C and D. (Section 6 will expand the following analysis to 3-bit and multiple bit changing mutation operators.)

This section will show that the Laplacian of the connectivity graph for k order parity (Δ_k) can be recursively created from two Laplacian for (k-1) parity and two Laplacians for the k-2 Hamming graph (H_{k-2}). I.e:

$$\Delta_k = \left(\begin{array}{c|c} (k-1)I + \Delta_{k-1} & H_{k-2} - (k-1)I \\ \hline H_{k-2} - (k-1)I & (k-1)I + \Delta_{k-1} \end{array} \right) \quad (2)$$

Where I is the identity matrix (of the appropriate dimensions). $(k-1)I$ is included in the two on-diagonal sub-matrices so that all the diagonal elements are $(k-1) + \frac{1}{2}(k-1)(k-2) = \frac{1}{2}k(k-1)$ as required. Since H (like Δ) is a graph Laplacian, its rows sum to zero. By subtracting $(k-1)I$ from the off diagonal matrices we ensure all the rows in $(k-1)I + \Delta_{k-1}|H_{k-2} - (k-1)I$ also sum to zero. (When we look in detail at H_{k-2} we will see that $-(k-1)I$ is exactly the adjustment needed for the Hamming cube adjacency matrix.)

Label the rows and columns of Δ_k with 2^{k-1} integers. These are given by the first 2^{k-1} integers starting from zero plus a k^{th} bit. This most significant bit is set (or cleared) to ensure the number of bits set in the row label is even. Order the rows/columns by the lower k-1 bits, ignoring the top (parity) bit. Elements of Δ are -1 if there are exactly two bits different in the row and column labels. Diagonal elements are $\frac{1}{2}k(k-1)$ and all other elements are zero.

Divide Δ into four equal sub-matrices. Half of Δ is made of the rows whose k-1^{th} bit is zero. The other half is made of the rows whose k-1^{th} bit is one. (Similarly for the columns). In the two on-diagonal sub-matrices the k-1^{th} bit in both the rows and the columns is the same.

If an element of Δ in the on-diagonal sub-matrices is -1, then this means that there are exactly two bits that differ in its row and column labels but the differing bits cannot include the k-1^{th} bit. If we remove the k-1^{th} bit, this is the condition for Δ_{k-1} to be -1. (We have to tidy up the labels by not just removing the k-1^{th} bit but also the k^{th} bits and recalculating the top, parity, bit for the k-2 bits.) As mentioned above, Δ_{k-1} has diagonal elements of $\frac{1}{2}(k-1)(k-2)$ so k-1 has to be added to them to convert the diagonal elements of Δ_{k-1} to those for Δ_k.

In the off diagonal sub-matrices of Δ, either the k-1^{th} bit of the row's label is zero and the k-1^{th} bit of the column's label is one or vice versa. I.e. the row and column's label already differs in one bit. If an element of Δ in the off-diagonal sub-matrices is -1, then, excluding the k-1^{th} and k^{th} bits, its row and column labels must differ by exactly one bit. (Alternatively the k-1^{th} and k^{th} bits both differ and the other k-2 bits are the same.) Ignoring the top two bits for a moment, we see that Δ being -1 in the off diagonals is exactly the same as the Hamming distance between two k-2 bit strings being one. That is, if H_{k-2} is -1 so too are the corresponding off diagonal elements in Δ. The only other non-zero elements of H are on the diagonal.

Since a k-bit string has k neighbours which differ by exactly one bit, the diagonal elements of H are k. I.e., the diagonal elements of H_{k-2} are k-2. These elements correspond to the lower k-2 bits of the row and column labels of Δ being the same. In Δ these elements are -1 (rather than k-2) since the two top bits can be simultaneously changed without changing the lower k-2 bits. Subtracting $k-1$ from the diagonal elements of H_{k-2} (i.e. from k-2) gives -1. Which is the value of the corresponding element in Δ_k. Hence we have proved Equation 2.

Δ_{k-1} can be defined in terms of Δ_{k-2} and H_{k-3} and so on. The base cases are: Δ_2 and H_1 which are both $\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$.

5.6 Eigen Analysis of Parity's Landscape

Using the recursive decomposition of Δ given in the previous section, we shall show if e_{k-1} is an eigenvector of Δ_{k-1} then $e_{k-1}|e_{k-1}$ and $e_{k-1}|-e_{k-1}$ are eigenvectors of Δ_k. It turns out the Walsh functions [24] are eigenvectors of both the Laplacian of the Hamming neighbourhood H and of the that of the Parity neighbourhood Δ. (Indeed, as Section 6 will prove, they are also eigenvectors of the Laplacians of many bit flip mutations.) The eigenvalues of Δ will be given at the end of this section.

$e_k\Delta_k =$

$$(e_{k-1}|\pm e_{k-1}) \left(\begin{array}{c|c} (k-1)I + \Delta_{k-1} & H_{k-2} - (k-1)I \\ \hline H_{k-2} - (k-1)I & (k-1)I + \Delta_{k-1} \end{array} \right) =$$

$$\begin{pmatrix} (k-1)e_{k-1} + \lambda e_{k-1} & \pm e_{k-1}H_{k-2} \mp (k-1)e_{k-1} \\ e_{k-1}H_{k-2} - (k-1)e_{k-1} \pm & (k-1)e_{k-1} \pm \lambda e_{k-1} \end{pmatrix} \quad (3)$$

Start with the base case (k=2): $\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$ has eigenvectors e_{2+}=(1,1) (eigenvalue 0) and e_{2-}=(1,-1) (eigenvalue 2). Notice that these are eigenvectors of both Δ_2 and H_1. So Equation 3 for k=3 becomes

$(e_2|\pm e_2)\Delta_3 =$

$$\begin{pmatrix} (k-1)e_{k-1} + \lambda e_{k-1} & \pm & \lambda e_{k-1} \mp (k-1)e_{k-1} \\ \lambda e_{k-1} - (k-1)e_{k-1} & \pm & (k-1)e_{k-1} \pm \lambda e_{k-1} \end{pmatrix} =$$

$$\begin{pmatrix} 2e_2 + \lambda e_2 & \pm & \lambda e_2 \mp 2e_2 \\ \lambda e_2 - 2e_2 & \pm & 2e_2 \pm \lambda e_2 \end{pmatrix} =$$

$$\begin{pmatrix} 2\lambda e_2 \\ 2\lambda e_2 \end{pmatrix} \text{ and } \begin{pmatrix} 4e_2 \\ -4e_2 \end{pmatrix}$$

That is $(e_2|e_2)$ are eigenvectors of Δ_3 (with eigenvalue 2λ) and so too are $(e_2|-e_2)$ (with eigenvalue 4). So Δ_3 has four eigenvectors:

$$\begin{array}{llrrr} e_{3++} & =(1, & 1, & 1, & 1) \\ e_{3+-} & =(1, & 1, & -1, & -1) \\ e_{3-+} & =(1, & -1, & 1, & -1) \\ e_{3--} & =(1, & -1, & -1, & 1) \end{array}$$

with corresponding eigenvalues: $\lambda_{3++} = 2\lambda_{2+} = 0$, $\lambda_{3+-} = 2\lambda_{2-} = 4$, $\lambda_{3-+} = 4$ and $\lambda_{3--} = 4$. We can see that the four eigenvectors form an orthonormal set and so this is a complete eigen description of Δ_3. The eigenvectors of Δ_3, e_3 are the Walsh basis (on 2 bits).

It can be shown that the Walsh basis (on k bits) are eigenvectors of the Hamming cube H_k, with eigenvalues $2i$ with multiplicity C_i^k for $0 \leq i \leq k$. (See, for example, Dr. Daniel Spielman's lecture notes (Lecture five, 16 September 2009 `http://www.cs.yale.edu/homes/spielman/561/lect05-09.pdf`) or [2].) Therefore the eigenvectors of Δ_3 are also eigenvectors of H_2 (albeit with different eigenvalues). We have e_3 are the Walsh basis therefore $e_3|e_3$ and $e_3|-e_3$ are the Walsh basis (3 bits). Let this hold for any higher order. I.e. $e_{k-1}|e_{k-1}$ and $e_{k-1}|-e_{k-1}$ are the Walsh basis on k-1 bits. Then Equation 3 becomes:

$$\begin{pmatrix} (k-1)e_{k-1} + \lambda_{k-1}e_{k-1} \pm & 2ie_{k-1} \mp (k-1)e_{k-1} \\ 2ie_{k-1} - (k-1)e_{k-1} & \pm (k-1)e_{k-1} \pm \lambda_{k-1}e_{k-1} \end{pmatrix} =$$

$$\begin{pmatrix} (2i + \lambda_{k-1})e_{k-1} \\ (2i + \lambda_{k-1})e_{k-1} \end{pmatrix} \text{ and } \begin{pmatrix} (2(k-1) - 2i + \lambda_{k-1})e_{k-1} \\ -(2(k-1) - 2i + \lambda_{k-1})e_{k-1} \end{pmatrix}$$

Table 1: Eigenvalues of Laplacian of Parity's landscape graph. The subscripts are the multiplicity of the Δ eigenvalues. Last column is the number of distinct eigenvalues. The total number of eigenvalues and eigenvectors is 2^{k-1}. (Catalogued as A176296 in the on-line encyclopedia of integer sequences.)

$k=2$	0_1	2_1					2
$k=3$	0_1	4_3					2
$k=4$	0_1	6_4	8_3				3
$k=5$	0_1	8_5	12_{10}				3
$k=6$	0_1	10_6	16_{15}	18_{10}			4
$k=7$	0_1	12_7	20_{21}	24_{35}			4
$k=8$	0_1	14_8	24_{28}	30_{56}	32_{35}		5
$k=9$	0_1	16_9	28_{36}	36_{84}	40_{126}		5
$k=10$	0_1	18_{10}	32_{45}	42_{120}	48_{210}	50_{126}	6
$k=11$	0_1	20_{11}	36_{55}	48_{165}	56_{330}	60_{462}	6
				$\vdots$			
$k=17$	0_1	32_{17}	60_{136}	84_{680}	104_{2380}	$\cdots\ 144_{24310}$	9

Where $2i$ is an eigenvalue of H_{k-2}. So $0 \leq i \leq k-2$. $2i$ and λ_{k-1} are related since they are the eigenvalues of H_{k-2} and Δ_{k-1} for the same eigenvector e_{k-1}.

That is $e_{k-1}|e_{k-1}$ are indeed eigenvectors of Δ_k (with eigenvalues $2i + \lambda_{k-1}$) and so too are $e_{k-1}| - e_{k-1}$ (with eigenvalues $2(k-1) - 2i + \lambda_{k-1}$). Notice since $e_{k-1}| \pm e_{k-1}$ are the Walsh basis, they form a complete orthogonal set of eigenvectors for Δ_k. The eigenvalues of Δ are not as elegant as those of the Hamming cube H but can be rapidly calculated, O(k^2). The first values are given in Table 1. Notice there are rather fewer distinct values than for the Hamming cube. In Section 5.9 we will prove that Δ_k has $k/2$+1 distinct eigenvalues.

The multiplicities shown as subscripts in Table 1 are similar to Pascal's triangle. (I.e. the eigenvalue multiplicities of the Hamming cube's Laplacian.) Firstly they sum to 2^{k-1} in each row. Also, except for the largest eigenvalue, each multiplicity is the sum of the multiplicities immediately above and to the left in the previous row. The multiplicity of the largest eigenvalue depends upon whether k is odd or even. If k is even, the multiplicity of the largest eigenvalue is the same as that in the previous row. If k odd, it is the sum of the multiplicity in the previous row to the left plus *twice* the multiplicity of the largest eigenvalue for k-1.

5.7 Largest Eigenvalue of Double Bit Flip Graph Laplacian

Since Δ is a Laplacian and hence its rows always sum to zero, its smallest eigenvalue is always zero (with multiplicity one). The smallest non-zero eigenvalue corresponds to the lowest Walsh function and has the value $2k$-2. Saying in closed form which Walsh functions have the largest eigenvalue is not straight forward. However we can 1) establish an upper bound on the largest eigenvalue and 2) give a stochastic estimate for large k.

5.7.1 Upper Bound on Largest Eigenvalue of Double Bit Flip Graph Laplacian

Depending upon the Walsh function chosen, an eigenvalue of Δ_k is either $\lambda_{H_{k-2}} + \lambda_{k-1}$ or $2(k-1) - \lambda_{H_{k-2}} + \lambda_{k-1}$. We know $\lambda_{H_{k-2}}$ cannot exceed $2k-4$ or be smaller than zero. Therefore the largest eigenvalue cannot exceed $2k-2+\max(\lambda_{k-1})$. We know $\max(\lambda_2) = 2$. So (for $k > 2$)

$$\begin{aligned}
\lambda_k &\leq \sum_{i=3}^{i=k} 2i - 2 + 2 \\
\lambda_k &\leq \sum_{i=3}^{i=k} 2i \\
\lambda_k &\leq -2 - 4 + 2\sum_{i=0}^{i=k} i \\
\lambda_k &\leq -2 - 4 + 2k(k+1)/2
\end{aligned}$$

Rearranging gives $\lambda_k \leq k(k+1) - 6$. However it appears the actual value is $\lceil (k-1)(k+1)/2 \rceil$.

5.7.2 Long Bit String Estimate of the Largest Eigenvalue of the 2-bit Flip Graph Laplacian

For any unit vector u the length of Δu will be no bigger than the largest eigenvalue. Choose a random direction. I.e. let v be a vector of 2^{k-1} components each of which is either +1 or -1, chosen uniformly at random. $|v|^2 = 2^{k-1}$ hence $|v| = 2^{(k-1)/2}$ (Eventually we will normalise v to be of unit length by dividing by $|v|$.) The first element of Δv is typical of them all.

$$\Delta v(1) = \pm\frac{1}{2}k(k-1) \qquad \underbrace{\pm 1 \pm \cdots\cdots\cdots \pm 1}_{\frac{1}{2}k(k-1) \text{ terms with random signs}}$$

The square of the length of Δv is given by summing the squares of each of its components in the usual way. Since the elements of v were randomly chosen, each of the elements of Δv (i.e. $\Delta v(i)$) are independent and identically distributed and therefore $|\Delta v(i)|^2$ are also i.i.d. Thus the expected value of $|\Delta v|^2$ is $2^{k-1}\times$ the expected value of any of them, e.g. the first $|\Delta v(1)|^2$. The expected length of $\Delta v(1)$ is approximately $\frac{1}{2}k(k-1)$. (The random signs ensure on average the following terms come to near zero and for large k the sum is dominated by the first (largest) term.) Thus the expected value of $|\Delta v|^2$ is $|\frac{1}{2}k(k-1)|^2 2^{k-1}$ and that of $|\Delta v| = \frac{1}{2}k(k-1)2^{(k-1)/2}$. Taking the ratio $\frac{|\Delta v|}{|v|}$ gives $\frac{1}{2}k(k-1)$ as an upper bound on all the eigenvalues. For large k this will become a tight bound on the largest eigenvalue.

Note the exact bound appears to be within a factor of two of the apparent value, whereas the stochastic upper bound appears to be increasingly tight as k increases. Finally note the eigenvalues of parity's Δ are a factor of k bigger than those of the Hamming cube.

5.8 Elementary Landscape Roughness and the Eigenvalues of the Graph Laplacian

Recall $d = \frac{1}{2}k(k-1)$ so the stochastic bound is needed to ensure $\frac{\lambda}{d} \leq 1$ for all cases (cf. Equation 1). Higher order Walsh functions are considered to be more rugged, since their sign changes more often than lower order Walsh functions. (Rothlauf points out [16, p27] that Walsh order is not a universal indicator of problem difficulty.) Elementary landscapes generated by higher Walsh basis functions have higher eigenvalues than those corresponding to lower order Walsh functions. Using Equation 1, we can see the higher an elementary landscape's eigenvalue the further each point is from the average of its neighbours. Thus we can view $\frac{\lambda}{d}$ as another measure of landscape ruggedness. The larger it is, the less each point tells us about the (average) fitness of its neighbours.

In other regular elementary landscapes eigenvalues can exceed the number of neighbours. I.e. $\frac{\lambda}{d}$ can exceed 1. For example in the Hamming cube the largest eigenvalue of H_k is $2k$ and each node has k neighbours. (So $0 \leq \lambda_H/d_H \leq 2$.) Whitley *et at.* [26, p589] equates $\frac{\lambda}{d} > 1$ with rugged elementary landscapes. Whereas they suggests if $0 \leq \frac{\lambda}{d} \leq 1$ the elementary landscape is smooth. ([26] uses a constant rather than referring to λ as an eigenvalue.)

We get the same conclusion if we use Whitley's [28] component based model of elementary landscapes. This treats their fitness as being composed of components which are added to and removed from the current trial solution as the search process moves from a point to one of its neighbours. [25, p383] gives $\frac{\lambda}{d} = p_1 + p_2$. Where p_1 is the proportion of components of $f(x)$ that change when we move away from x. p_2 is the proportion of components not included in $f(x)$ that change when we move away from x. Thus we should expect a small value of $p_1 + p_2$ to give a smooth landscape. When $p_1 + p_2$ approaches one, most of the components are being changed at each move, so we expect a more rugged landscape, in keeping with the previous paragraph.

5.9 Number of Distinct Eigenvalues and Parity's Graph Diameter

A graph's diameter is the maximum distance between any two nodes in the graph. (Where the distance is the smallest number of edges that have to be traversed to go between the nodes.) Parity is symmetric so the longest distance between any pair of nodes, is the same as the longest distance between the origin and any node. This is the minimum number of pairs of bit flips between k-1 zeros and the binary string of the target node. This is simply $bc(\text{target})/2$, where bc is the number of 1s (the bit count). Obviously the worst case is when all bits are one. Hence Δ's graph diameter is $k/2$. Whereas the diameter of the Hamming cube is k-1.

Reeves [14, page 598] says the number of distinct eigenvalues is the graph diameter plus one. We can see, from the last column of Table 1, that the number of distinct eigenvalues is indeed $k/2$+1.

5.10 Number of Distinct Elementary Landscapes and Walsh Analysis

Since the eigenvectors form an orthogonal set, any vector, including the fitness f vector, can be represented by its components projected onto the eigenvectors. Remembering Section 5.3, each eigenvector of Δ represents an elementary landscape, so fitness can be represented as a sum of elementary landscapes. One for each eigenvector where it has a non-zero projection. As the Walsh basis functions are eigenvectors of Δ projecting the fitness f vector onto these eigenvectors is equivalent to the Walsh analysis of f. Since f is 1 followed by $2^{k-1} - 1$ zeros it has 2^{k-1} non-zero Walsh coefficients. That is all its Walsh coefficients are non-zero (actually they are all equal to $2^{-(k-1)}$). However eigenvectors with the same eigenvalue form sub-spaces where any linear combination of these eigenvectors is also an eigenvector. Hence any vector can be represented as a sum of eigenvectors, one for each subspace where it has non-zero projection. We know that Parity's fitness vector has non-zero projection into each subspace so all $k/2 + 1$ subspaces must be used. That is, Parity's fitness function can be expressed as $k/2 + 1$ elementary fitness landscapes. Indeed $k/2 + 1$ is an upper limit on the number of elementary landscapes needed to represent any fitness function (when we use only a 2-bit flip mutation operator).

6. HIGHER BIT FLIP LANDSCAPES

6.1 Construction of 3-Bit Flip Laplacian

We can extend the construction used in Section 5.5 to consider the landscape formed by a mutation operator which flips exactly three bits.

$$\Delta_{3,l} = \left(\begin{array}{c|c} (C_3^l - C_3^{l-1})I + \Delta_{3,l-1} & \Delta_{2,l-1} - (C_3^l - C_3^{l-1})I \\ \hline \Delta_{2,l-1} - (C_3^l - C_3^{l-1})I & (C_3^l - C_3^{l-1})I + \Delta_{3,l-1} \end{array} \right) \tag{4}$$

We use $\Delta_{n,l}$ to refer to the Laplacian of the graph formed by flipping exactly n bits in strings of length l. Note, unlike for parity in Section 5, we allow the graph to be disjoint. This simplifies the analysis a little. For example, all strings are of length l rather than being shorter if parts of the search space are inaccessible.

Unlike the double flip operator, every point in the search space can be reached eventually by mutating exactly three bits. Therefore, when using Equation 4, we have to use a version of the Laplacian which includes the half of the search space which (starting from the origin) was previously inaccessible. E.g. for three bits the new Laplacian for flipping two bits (cf. Δ_3 Section 5.3) is:

$$\Delta_{2,3} = \begin{array}{c|cccccccc} & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\ \hline 000 & 3 & 0 & 0 & -1 & 0 & -1 & -1 & 0 \\ 001 & 0 & 3 & -1 & 0 & -1 & 0 & 0 & -1 \\ 010 & 0 & -1 & 3 & 0 & -1 & 0 & 0 & -1 \\ 011 & -1 & 0 & 0 & 3 & 0 & -1 & -1 & 0 \\ 100 & 0 & -1 & -1 & 0 & 3 & 0 & 0 & -1 \\ 101 & -1 & 0 & 0 & -1 & 0 & 3 & -1 & 0 \\ 110 & -1 & 0 & 0 & -1 & 0 & -1 & 3 & 0 \\ 111 & 0 & -1 & -1 & 0 & -1 & 0 & 0 & 3 \end{array}$$

$$\Delta_{3,3} = \begin{array}{c|cccccccc} & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\ \hline 000 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 001 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 010 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 011 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 100 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 101 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 110 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 111 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}$$

$$\Delta_{3,4} = \begin{array}{r|cccccccccccccccc}
0000 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & 0 \\
0001 & 0 & 4 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & -1 \\
0010 & 0 & 0 & 4 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 \\
0011 & 0 & 0 & 0 & 4 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 \\
0100 & 0 & 0 & 0 & -1 & 4 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & -1 \\
0101 & 0 & 0 & -1 & 0 & 0 & 4 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \\
0110 & 0 & -1 & 0 & 0 & 0 & 0 & 4 & 0 & -1 & 0 & 0 & -1 & 0 & -1 & 0 & 0 \\
0111 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & -1 & -1 & 0 & -1 & 0 & 0 & 0 \\
1000 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
1001 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & -1 & 0 & 4 & 0 & 0 & 0 & 0 & -1 & 0 \\
1010 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 4 & 0 & 0 & -1 & 0 & 0 \\
1011 & -1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 4 & -1 & 0 & 0 & 0 \\
1100 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 4 & 0 & 0 & 0 \\
1101 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 4 & 0 & 0 \\
1110 & -1 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 4 & 0 \\
1111 & 0 & -1 & -1 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 4
\end{array}$$

Using $\Delta_{2,3}$ and $\Delta_{3,3}$ we can easily see that Equation 4 holds for $\Delta_{3,4}$:

$$\Delta_{3,4} = \left(\begin{array}{c|c} 3I + \Delta_{3,3} & \Delta_{2,3} - 3I \\ \hline \Delta_{2,3} - 3I & 3I + \Delta_{3,3} \end{array} \right)$$

To show Equation 4 holds in general we reuse the argument given in Section 5.5. Essentially in the on diagonal quadrants of $\Delta_{3,l}$ the most significant bits of the node labels are the same. I.e. either both 0 or both 1. Thus in these two quadrants, where the node labels differ by exactly three bits (a "-1" element) the different bits must be in the lower $l-1$ bits. Which is exactly the condition for $\Delta_{3,l-1}$ to also have a -1 element. In the off diagonal quadrants, the most significant bits are different. Thus any -1 element must correspond to exactly two bits being different in the lower $l-1$ bits of the search space labels. This is exactly the condition of $\Delta_{2,l-1}$ to also have a -1 element. It only remains to fix up the diagonal elements of the four quadrants.

The diagonal elements in the three bit mutation's Laplacians are C_3^l (where the bit string length is l). Correspondingly the diagonal elements of $\Delta_{3,l-1}$ are C_3^{l-1}. So, in equation 4, we need to add $C_3^l - C_3^{l-1}$ to the main diagonal.

$$\begin{aligned}
C_3^l - C_3^{l-1} &= \frac{l(l-1)!}{3!(l-3)!} - \frac{(l-1)!}{3!(l-4)!} \\
&= \frac{(l-1)!}{3!(l-3)!} \times (l-(l-3)) \\
&= \frac{(l-1)!}{3!(l-3)!} \times 3 \\
&= \frac{(l-1)!}{2!(l-1-2)!} \\
&= C_2^{l-1}
\end{aligned}$$

Having added to C_2^{l-1} to the two on-diagonal quadrants, to ensure every row of $\Delta_{3,l}$ continues to sum to zero, we must subtract C_2^{l-1} from the diagonal of the two off-diagonal quadrants. Notice, since the diagonal elements of $\Delta_{2,l-1}$ are C_2^{l-1}, the diagonals of the off diagonal quadrants are zero.

6.2 Construction of n-Bit Flip Laplacian

We can extend the construction used in the previous section to consider the landscape formed by a mutation operator which flips exactly n bits.

We start by simplifying the diagonal terms. Again we need to add a multiple of I to the main diagonal and subtract it in the remaining two quadrants. (See previous section.) For the n bit Laplacian we must add $C_n^l - C_n^{l-1}$ to ensure the diagonal of Δ_n^{l-1} is brought up to that of Δ_n^l

$$\begin{aligned}
C_n^l - C_n^{l-1} &= \frac{l(l-1)!}{n!(l-n)!} - \frac{(l-1)!}{n!(l-1-n)!} \\
&= \frac{(l-1)!}{n!(l-n)!} \times (l-(l-n)) \\
&= \frac{(l-1)!}{n!(l-n)!} \times n \\
&= \frac{(l-1)!}{(n-1)!(l-1-(n-1))!} \\
&= C_{n-1}^{l-1}
\end{aligned}$$

Notice again that this is equal to the diagonal elements of $\Delta_{n,l-1}$, and so in general the diagonal elements of the off diagonal quadrants corresponding to the lower order mutation operator acting on the shorter bit string are zero. This is correct, since n-bit flip mutation cannot simply flip the most significant bit (which correspond to changing only one bit).

$$\Delta_{n,l} = \left(\begin{array}{c|c} C_{n-1}^{l-1} I + \Delta_{n,l-1} & \Delta_{n-1,l-1} - C_{n-1}^{l-1} I \\ \hline \Delta_{n-1,l-1} - C_{n-1}^{l-1} I & C_{n-1}^{l-1} I + \Delta_{n,l-1} \end{array} \right) \qquad (5)$$

To give a concrete example, consider the Laplacian for flipping exactly five bits. The example shows it can be decomposed into two lower order five bit and two four bit Laplacians (plus C_4^{l-1} multiples of the identity matrix I). To make the symmetries clearer, we have only printed the non-zero elements.

$$\Delta_{4,5} = \begin{array}{r|cccccccccccccccccccccccccccccccc}
00000 & 5&&&&&&&&&&&&&&&-1&&&&&&&&-1&&&&-1&&-1&-1& \\
00001 & &5&&&&&&&&&&&&&-1&&&&&&&&-1&&&&-1&&-1&&&-1 \\
00010 & &&5&&&&&&&&&&&-1&&&&&&&&-1&&&&-1&&&-1&&&-1 \\
00011 & &&&5&&&&&&&&&-1&&&&&&&&-1&&&&-1&&&&&-1&-1& \\
00100 & &&&&5&&&&&&&-1&&&&&&&&-1&&&&&&-1&-1&&&&&-1 \\
00101 & &&&&&5&&&&&-1&&&&&&&&-1&&&&&&-1&&&-1&&&-1& \\
00110 & &&&&&&5&&&-1&&&&&&&&-1&&&&&&&-1&&&-1&&-1&& \\
00111 & &&&&&&&5&-1&&&&&&&&-1&&&&&&&&&-1&-1&&-1&&& \\
01000 & &&&&&&&-1&5&&&&&&&&&&&-1&&-1&-1&&&&&&&&&-1 \\
01001 & &&&&&&-1&&&5&&&&&&&&&-1&&-1&&&-1&&&&&&&-1& \\
01010 & &&&&&-1&&&&&5&&&&&&&-1&&&-1&&&-1&&&&&&-1&& \\
01011 & &&&&-1&&&&&&&5&&&&&-1&&&&&-1&-1&&&&&&-1&&& \\
01100 & &&&-1&&&&&&&&&5&&&&&-1&-1&&&&&-1&&&&-1&&&& \\
01101 & &&-1&&&&&&&&&&&5&&&-1&&&-1&&&-1&&&&-1&&&&& \\
01110 & &-1&&&&&&&&&&&&&5&&-1&&&-1&&-1&&&&-1&&&&&& \\
01111 & -1&&&&&&&&&&&&&&&5&&-1&-1&&-1&&&&-1&&&&&&& \\
10000 & &&&&&&&-1&&&&-1&&-1&-1&&5&&&&&&&&&&&&&&&-1 \\
10001 & &&&&&&-1&&&&-1&&-1&&&-1&&5&&&&&&&&&&&&&-1& \\
10010 & &&&&&-1&&&&-1&&&-1&&&-1&&&5&&&&&&&&&&&-1&& \\
10011 & &&&&-1&&&&-1&&&&&-1&-1&&&&&5&&&&&&&&&-1&&& \\
10100 & &&&-1&&&&&&-1&-1&&&&&-1&&&&&5&&&&&&&-1&&&& \\
10101 & &&-1&&&&&&-1&&&-1&&&-1&&&&&&&5&&&&&-1&&&&& \\
10110 & &-1&&&&&&&-1&&&-1&&-1&&&&&&&&&5&&&-1&&&&&& \\
10111 & -1&&&&&&&&&-1&-1&&-1&&&&&&&&&&&5&-1&&&&&&& \\
11000 & &&&-1&&-1&-1&&&&&&&&&-1&&&&&&&&-1&5&&&&&&& \\
11001 & &&-1&&-1&&&-1&&&&&&&-1&&&&&&&&-1&&&5&&&&&& \\
11010 & &-1&&&-1&&&-1&&&&&&-1&&&&&&&&-1&&&&&5&&&&& \\
11011 & -1&&&&&-1&-1&&&&&&-1&&&&&&&&-1&&&&&&&5&&&& \\
11100 & &-1&-1&&&&&-1&&&&-1&&&&&&&&-1&&&&&&&&&5&&& \\
11101 & -1&&&-1&&&-1&&&&-1&&&&&&&&-1&&&&&&&&&&&5&& \\
11110 & -1&&&-1&&-1&&&&-1&&&&&&&&-1&&&&&&&&&&&&&5& \\
11111 & &-1&-1&&-1&&&&-1&&&&&&&&-1&&&&&&&&&&&&&&&5
\end{array}$$

$$\Delta_{5,5} =$$

$$\begin{array}{l|cccccccccccccccccccccccccccccccc}
00000 & 1 &&&&& &&&&& &&&&& &&&&& &&&&& &&&&& & -1 \\
00001 && 1 &&&&& &&&&& &&&&& &&&&& &&&&& &&&& -1 \\
00010 &&& 1 &&&&& &&&&& &&&&& &&&&& &&&&& && -1 \\
00011 &&&& 1 &&&&& &&&&& &&&&& &&&&& &&&&& -1 \\
00100 &&&&& 1 &&&&& &&&&& &&&&& &&&&& &&& -1 \\
00101 &&&&& & 1 &&&&& &&&&& &&&&& &&&&& & -1 \\
00110 &&&&& && 1 &&&&& &&&&& &&&&& &&&& -1 \\
00111 &&&&& &&& 1 &&&&& &&&&& &&&&& && -1 \\
01000 &&&&& &&&& 1 &&&&& &&&&& &&&&& -1 \\
01001 &&&&& &&&&& 1 &&&&& &&&&& &&& -1 \\
01010 &&&&& &&&&& & 1 &&&&& &&&&& & -1 \\
01011 &&&&& &&&&& && 1 &&&&& &&&& -1 \\
01100 &&&&& &&&&& &&& 1 &&&&& && -1 \\
01101 &&&&& &&&&& &&&& 1 &&&&& -1 \\
01110 &&&&& &&&&& &&&&& 1 &&& -1 \\
01111 &&&&& &&&&& &&&&& & 1 & -1 \\
10000 &&&&& &&&&& &&&&& & -1 & 1 \\
10001 &&&&& &&&&& &&&&& -1 &&& 1 \\
10010 &&&&& &&&&& &&&& -1 &&&&& 1 \\
10011 &&&&& &&&&& &&& -1 &&&&& && 1 \\
10100 &&&&& &&&&& && -1 &&&&& &&&& 1 \\
10101 &&&&& &&&&& & -1 &&&&& &&&&& & 1 \\
10110 &&&&& &&&&& -1 &&&&& &&&&& &&& 1 \\
10111 &&&&& &&&& -1 &&&&& &&&&& &&&&& 1 \\
11000 &&&&& &&& -1 &&&&& &&&&& &&&&& && 1 \\
11001 &&&&& && -1 &&&&& &&&&& &&&&& &&&& 1 \\
11010 &&&&& & -1 &&&&& &&&&& &&&&& &&&&& & 1 \\
11011 &&&&& -1 &&&&& &&&&& &&&&& &&&&& &&& 1 \\
11100 &&&& -1 &&&&& &&&&& &&&&& &&&&& &&&&& 1 \\
11101 &&& -1 &&&&& &&&&& &&&&& &&&&& &&&&& && 1 \\
11110 && -1 &&&&& &&&&& &&&&& &&&&& &&&&& &&&& 1 \\
11111 & -1 &&&&& &&&&& &&&&& &&&&& &&&&& &&&&& & 1
\end{array}$$

Using $\Delta_{4,5}$ and $\Delta_{5,5}$ we see that Equation 5 holds for $\Delta_{5,6}$:

$$\Delta_{5,6} = \left(\begin{array}{c|c} 5I + \Delta_{5,5} & \Delta_{4,5} - 5I \\ \hline \Delta_{4,5} - 5I & 5I + \Delta_{5,5} \end{array} \right)$$

The proof of Equation 5 follows that for two and three bit flip mutation (Sections 5.5 and 6.1). In the on diagonal quadrants of $\Delta_{n,l}$ the most significant bits of the node labels are the same, thus in these two quadrants, where there is a -1 element, the node labels must differ in exactly n positions. Since the most significant positions are the same, the n differences must be in the lower part of the node labels. Which is the condition for $\Delta_{n,l-1}$ to have a -1 element. Whereas in the two off diagonal quadrants, the top positions do differ, so each -1 element means there are n-1 differences in the lower parts of the node labels. Which is the condition for $\Delta_{n-1,l-1}$ to have a -1 element. The only other non-zero elements are the diagonals, which (at the beginning of this section) we showed need to be adjusted by $\pm C^{l-1}_{n-1}$.

The base case for flipping exactly n bits is the set of strings of n bits. Here mutation is highly constrained and all it can do is move between each string and its complement. So although there are 2^n strings in the fitness landscape, with n-bit mutation, the landscape falls into 2^{n-1} parts. Each part contains exactly two string. Each part is inaccessible from all the others. Thus $\Delta_{n,n}$ is the $2^n \times 2^n$ sparse symmetric array with exactly one -1 per row (along the trailing diagonal) and 1 on the main diagonal (see $\Delta_{3,3}$ and $\Delta_{5,5}$). Starting from the set of $\Delta_{n,n}$, Equation 5 can be used recursively to construct the Laplacian for mutation of binary strings by flipping exactly n bits.

6.3 Eigen Analysis of n-bit Mutation

Re-using the eigen analysis for the case of flipping exactly two bits (given in Section 5.6) and the recursive decomposition of $\Delta_{n,l}$ given in the previous section, we shall show the Walsh basis are also eigenvectors of the Laplacian describing n-bit mutation's landscape.

The Walsh basis are vectors of length 2^l. The j^{th} Walsh basis vector's components are: $\psi_j(x) = (-1)^{bc(j \wedge x)}$, where x is the index of the vector element (starting from 0) and bc is again the number of bits set to one. So $\psi_j(x)$ is -1 if $bc(j \wedge x)$ is odd and 1 if it is even.

The base cases for flipping n bits, $\Delta_{n,n}$, can be split into the sum of the identity matrix (of size $2^n \times 2^n$) and a matrix of the same size which is also zero's except for -1 along the reverse diagonal (e.g. $\Delta_{3,3}$ and $\Delta_{5,5}$). Call this matrix N. Thus $\Delta_{n,n} = I + N$. Multiplying a vector by N, reverses the order of its elements and then multiplies them by -1. Notice reversing the elements means replacing element x by element $\overline{x}$, where $\overline{x}$ is the bitwise complement of x. Hence $\psi_j(x)\Delta_{n,n} = \psi_j(x)(I+N) = \psi_j(x) - \psi_j(\overline{x}) = \psi_j(x) - (-1)^{bc(j \wedge \overline{x})}$.

Suppose j has t bits set. (I.e. $bc(j) = t$.) Only the bits of x which match these influence the calculation of $bc(j \wedge x)$. Suppose, in these positions, x has k bits set and m zeros. (Note $k = bc(j \wedge x)$ and $m = t - k$.) Then $\overline{x}$ has k zeros and m ones matching set bits in j. Hence $bc(j \wedge \overline{x}) = m$. $(-1)^{bc(j \wedge \overline{x})} = (-1)^m = (-1)^{t-k} = (-1)^t(-1)^{-k} = (-1)^t(-1)^k$ $= (-1)^t(-1)^{bc(j\wedge x)} = (-1)^{bc(j)}\psi_j(x)$. Hence $\psi_j(x)\Delta_{n,n} = \psi_j(x) - (-1)^{bc(j)}\psi_j(x) = 0$ or $2\psi_j(x)$. I.e. the Walsh basis vectors, $\psi_j(x)$, are eigenvectors of $\Delta_{n,n}$. The corresponding eigenvalue is either zero (if the number of bits set in j is even) or $\lambda = 2$ (if $bc(j)$ is odd). Remember ψ_j are orthogonal and there are 2^n of them. In other words, the Walsh basis form a complete set of eigenvectors for $\Delta_{n,n}$.

To show the Walsh basis are also eigenvectors of $\Delta_{n,l}$ we reuse the fact that the higher order Walsh basis can be constructed by concatenated each vector with itself or with -1 times itself. I.e., $(e_{l-1}|e_{l-1})$ and $(e_{l-1}|-e_{l-1})$. Using Equation 5:

$$e_l\Delta_{n,l} =$$

$$(e_{l-1}|\pm e_{l-1}) \left(\begin{array}{c|c} C^{l-1}_{n-1}I + \Delta_{n,l-1} & \Delta_{n-1,l-1} - C^{l-1}_{n-1}I \\ \hline \Delta_{n-1,l-1} - C^{l-1}_{n-1}I & C^{l-1}_{n-1}I + \Delta_{n,l-1} \end{array} \right)$$

$$= \left(\begin{array}{c} C^{l-1}_{n-1}e_{l-1} + \lambda_{n,l-1}e_{l-1} \pm \lambda_{n-1,l-1}e_{l-1} \mp C^{l-1}_{n-1}e_{l-1} \\ \lambda_{n-1,l-1}e_{l-1} - C^{l-1}_{n-1}e_{l-1} \pm C^{l-1}_{n-1}e_{l-1} \quad \pm\lambda_{n,l-1}e_{l-1} \end{array} \right)$$

$$= \left(\begin{array}{c} (\lambda_{n,l-1} + \lambda_{n-1,l-1})\, e_{l-1} \\ (\lambda_{n,l-1} + \lambda_{n-1,l-1})\, e_{l-1} \end{array} \right)$$

or

$$= \left(\begin{array}{c} \left(2C^{l-1}_{n-1} + \lambda_{n,l-1} - \lambda_{n-1,l-1}\right) e_{l-1} \\ -\left(2C^{l-1}_{n-1} + \lambda_{n,l-1} - \lambda_{n-1,l-1}\right) e_{l-1} \end{array} \right)$$

Note e_{l-1} is an eigenvector of both $\Delta_{n,l-1}$ (with eigenvalue denoted $\lambda_{n,l-1}$) and of $\Delta_{n-1,l-1}$ (with eigenvalue denoted $\lambda_{n-1,l-1}$). Therefore $(e_{l-1}|e_{l-1})$ is an eigenvector of $\Delta_{n,l}$ with eigenvalue

$$\lambda_{n,l} = \lambda_{n,l-1} + \lambda_{n-1,l-1} \tag{6}$$

and $(e_{l-1}|-e_{l-1})$ is another eigenvector of $\Delta_{n,l}$ (the higher Walsh vector) with eigenvalue

$$\lambda_{n,l} = 2C^{l-1}_{n-1} + \lambda_{n,l-1} - \lambda_{n-1,l-1}. \tag{7}$$

Since $\lambda_{n,n}$ is either 0 or 2, $\lambda_{n,l}$ will always be real, integer and even. Table 2 shows an example where Equations 6 and 7 are used recursively.

Table 3 gives some example eigenvectors of the graph Laplacians formed by various bit flip mutation operators. Notice that, apart from the zeroth order Walsh basis vector, no eigenvector has the same eigenvalue for all mutation operators.

$$\Delta_{5,6} = \begin{array}{r|cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc}
000000 &6&&&&&&& &&&&&&&& &&&&&&&& &&&&&&&&-1 &&&&&&&& &&&&&&&&-1 &&&&&&&&-1 &&&&-1&&-1&-1& \\
000001 &&6&&&&&& &&&&&&&& &&&&&&&& &&&&&&&-1& &&&&&&&& &&&&&&&-1& &&&&&&&-1& &&&-1&&-1&&&-1 \\
000010 &&&6&&&&& &&&&&&&& &&&&&&&& &&&&&&-1&& &&&&&&&& &&&&&&-1&& &&&&&&-1&& &&-1&&&-1&&&-1 \\
000011 &&&&6&&&& &&&&&&&& &&&&&&&& &&&&&-1&&& &&&&&&&& &&&&&-1&&& &&&&&-1&&& &-1&&&&&-1&-1& \\
000100 &&&&&6&&& &&&&&&&& &&&&&&&& &&&&-1&&&& &&&&&&&& &&&&-1&&&& &&&&-1&&&& &&-1&-1&&&&&-1 \\
000101 &&&&&&6&& &&&&&&&& &&&&&&&& &&&-1&&&&& &&&&&&&& &&&-1&&&&& &&&-1&&&&& &-1&&&-1&&&-1& \\
000110 &&&&&&&6& &&&&&&&& &&&&&&&& &&-1&&&&&& &&&&&&&& &&-1&&&&&& &&-1&&&&&& &-1&&&-1&&-1&& \\
000111 &&&&&&&&6 &&&&&&&& &&&&&&&& &-1&&&&&&& &&&&&&&& &-1&&&&&&& &-1&&&&&&& &&-1&-1&&-1&&& \\
001000 &&&&&&&& &6&&&&&&& &&&&&&&&-1 &&&&&&&& &&&&&&&&-1 &&&&&&&& &&&&-1&&-1&-1& &&&&&&&&-1 \\
001001 &&&&&&&& &&6&&&&&& &&&&&&&-1& &&&&&&&& &&&&&&&-1& &&&&&&&& &&&-1&&-1&&&-1 &&&&&&&-1& \\
001010 &&&&&&&& &&&6&&&&& &&&&&&-1&& &&&&&&&& &&&&&&-1&& &&&&&&&& &&-1&&&-1&&&-1 &&&&&&-1&& \\
001011 &&&&&&&& &&&&6&&&& &&&&&-1&&& &&&&&&&& &&&&&-1&&& &&&&&&&& &-1&&&&&-1&-1& &&&&&-1&&& \\
001100 &&&&&&&& &&&&&6&&& &&&&-1&&&& &&&&&&&& &&&&-1&&&& &&&&&&&& &&-1&-1&&&&&-1 &&&&-1&&&& \\
001101 &&&&&&&& &&&&&&6&& &&&-1&&&&& &&&&&&&& &&&-1&&&&& &&&&&&&& &-1&&&-1&&&-1& &&&-1&&&&& \\
001110 &&&&&&&& &&&&&&&6& &&-1&&&&&& &&&&&&&& &&-1&&&&&& &&&&&&&& &-1&&&-1&&-1&& &&-1&&&&&& \\
001111 &&&&&&&& &&&&&&&&6 &-1&&&&&&& &&&&&&&& &-1&&&&&&& &&&&&&&& &&-1&-1&&-1&&& &-1&&&&&&& \\
010000 &&&&&&&& &&&&&&&&-1 &6&&&&&&& &&&&&&&& &&&&&&&&-1 &&&&-1&&-1&-1& &&&&&&&& &&&&&&&&-1 \\
010001 &&&&&&&& &&&&&&&-1& &&6&&&&&& &&&&&&&& &&&&&&&-1& &&&-1&&-1&&&-1 &&&&&&&& &&&&&&&-1& \\
010010 &&&&&&&& &&&&&&-1&& &&&6&&&&& &&&&&&&& &&&&&&-1&& &&-1&&&-1&&&-1 &&&&&&&& &&&&&&-1&& \\
010011 &&&&&&&& &&&&&-1&&& &&&&6&&&& &&&&&&&& &&&&&-1&&& &-1&&&&&-1&-1& &&&&&&&& &&&&&-1&&& \\
010100 &&&&&&&& &&&&-1&&&& &&&&&6&&& &&&&&&&& &&&&-1&&&& &&-1&-1&&&&&-1 &&&&&&&& &&&&-1&&&& \\
010101 &&&&&&&& &&&-1&&&&& &&&&&&6&& &&&&&&&& &&&-1&&&&& &-1&&&-1&&&-1& &&&&&&&& &&&-1&&&&& \\
010110 &&&&&&&& &&-1&&&&&& &&&&&&&6& &&&&&&&& &&-1&&&&&& &-1&&&-1&&-1&& &&&&&&&& &&-1&&&&&& \\
010111 &&&&&&&& &-1&&&&&&& &&&&&&&&6 &&&&&&&& &-1&&&&&&& &&-1&-1&&-1&&& &&&&&&&& &-1&&&&&&& \\
011000 &&&&&&&&-1 &&&&&&&& &&&&&&&& &6&&&&&&& &&&&-1&&-1&-1& &&&&&&&&-1 &&&&&&&&-1 &&&&&&&& \\
011001 &&&&&&&-1& &&&&&&&& &&&&&&&& &&6&&&&&& &&&-1&&-1&&&-1 &&&&&&&-1& &&&&&&&-1& &&&&&&&& \\
011010 &&&&&&-1&& &&&&&&&& &&&&&&&& &&&6&&&&& &&-1&&&-1&&&-1 &&&&&&-1&& &&&&&&-1&& &&&&&&&& \\
011011 &&&&&-1&&& &&&&&&&& &&&&&&&& &&&&6&&&& &-1&&&&&-1&-1& &&&&&-1&&& &&&&&-1&&& &&&&&&&& \\
011100 &&&&-1&&&& &&&&&&&& &&&&&&&& &&&&&6&&& &&-1&-1&&&&&-1 &&&&-1&&&& &&&&-1&&&& &&&&&&&& \\
011101 &&&-1&&&&& &&&&&&&& &&&&&&&& &&&&&&6&& &-1&&&-1&&&-1& &&&-1&&&&& &&&-1&&&&& &&&&&&&& \\
011110 &&-1&&&&&& &&&&&&&& &&&&&&&& &&&&&&&6& &-1&&&-1&&-1&& &&-1&&&&&& &&-1&&&&&& &&&&&&&& \\
011111 &-1&&&&&&& &&&&&&&& &&&&&&&& &&&&&&&&6 &&-1&-1&&-1&&& &-1&&&&&&& &-1&&&&&&& &&&&&&&& \\
100000 &&&&&&&& &&&&&&&&-1 &&&&&&&&-1 &&&&-1&&-1&-1& &6&&&&&&& &&&&&&&& &&&&&&&& &&&&&&&&-1 \\
100001 &&&&&&&& &&&&&&&-1& &&&&&&&-1& &&&-1&&-1&&&-1 &&6&&&&&& &&&&&&&& &&&&&&&& &&&&&&&-1& \\
100010 &&&&&&&& &&&&&&-1&& &&&&&&-1&& &&-1&&&-1&&&-1 &&&6&&&&& &&&&&&&& &&&&&&&& &&&&&&-1&& \\
100011 &&&&&&&& &&&&&-1&&& &&&&&-1&&& &-1&&&&&-1&-1& &&&&6&&&& &&&&&&&& &&&&&&&& &&&&&-1&&& \\
100100 &&&&&&&& &&&&-1&&&& &&&&-1&&&& &&-1&-1&&&&&-1 &&&&&6&&& &&&&&&&& &&&&&&&& &&&&-1&&&& \\
100101 &&&&&&&& &&&-1&&&&& &&&-1&&&&& &-1&&&-1&&&-1& &&&&&&6&& &&&&&&&& &&&&&&&& &&&-1&&&&& \\
100110 &&&&&&&& &&-1&&&&&& &&-1&&&&&& &-1&&&-1&&-1&& &&&&&&&6& &&&&&&&& &&&&&&&& &&-1&&&&&& \\
100111 &&&&&&&& &-1&&&&&&& &-1&&&&&&& &&-1&-1&&-1&&& &&&&&&&&6 &&&&&&&& &&&&&&&& &-1&&&&&&& \\
101000 &&&&&&&&-1 &&&&&&&& &&&&-1&&-1&-1& &&&&&&&&-1 &&&&&&&& &6&&&&&&& &&&&&&&&-1 &&&&&&&& \\
101001 &&&&&&&-1& &&&&&&&& &&&-1&&-1&&&-1 &&&&&&&-1& &&&&&&&& &&6&&&&&& &&&&&&&-1& &&&&&&&& \\
101010 &&&&&&-1&& &&&&&&&& &&-1&&&-1&&&-1 &&&&&&-1&& &&&&&&&& &&&6&&&&& &&&&&&-1&& &&&&&&&& \\
101011 &&&&&-1&&& &&&&&&&& &-1&&&&&-1&-1& &&&&&-1&&& &&&&&&&& &&&&6&&&& &&&&&-1&&& &&&&&&&& \\
101100 &&&&-1&&&& &&&&&&&& &&-1&-1&&&&&-1 &&&&-1&&&& &&&&&&&& &&&&&6&&& &&&&-1&&&& &&&&&&&& \\
101101 &&&-1&&&&& &&&&&&&& &-1&&&-1&&&-1& &&&-1&&&&& &&&&&&&& &&&&&&6&& &&&-1&&&&& &&&&&&&& \\
101110 &&-1&&&&&& &&&&&&&& &-1&&&-1&&-1&& &&-1&&&&&& &&&&&&&& &&&&&&&6& &&-1&&&&&& &&&&&&&& \\
101111 &-1&&&&&&& &&&&&&&& &&-1&-1&&-1&&& &-1&&&&&&& &&&&&&&& &&&&&&&&6 &-1&&&&&&& &&&&&&&& \\
110000 &&&&&&&&-1 &&&&-1&&-1&-1& &&&&&&&& &&&&&&&&-1 &&&&&&&& &&&&&&&&-1 &6&&&&&&& &&&&&&&& \\
110001 &&&&&&&-1& &&&-1&&-1&&&-1 &&&&&&&& &&&&&&&-1& &&&&&&&& &&&&&&&-1& &&6&&&&&& &&&&&&&& \\
110010 &&&&&&-1&& &&-1&&&-1&&&-1 &&&&&&&& &&&&&&-1&& &&&&&&&& &&&&&&-1&& &&&6&&&&& &&&&&&&& \\
110011 &&&&&-1&&& &-1&&&&&-1&-1& &&&&&&&& &&&&&-1&&& &&&&&&&& &&&&&-1&&& &&&&6&&&& &&&&&&&& \\
110100 &&&&-1&&&& &&-1&-1&&&&&-1 &&&&&&&& &&&&-1&&&& &&&&&&&& &&&&-1&&&& &&&&&6&&& &&&&&&&& \\
110101 &&&-1&&&&& &-1&&&-1&&&-1& &&&&&&&& &&&-1&&&&& &&&&&&&& &&&-1&&&&& &&&&&&6&& &&&&&&&& \\
110110 &&-1&&&&&& &-1&&&-1&&-1&& &&&&&&&& &&-1&&&&&& &&&&&&&& &&-1&&&&&& &&&&&&&6& &&&&&&&& \\
110111 &-1&&&&&&& &&-1&-1&&-1&&& &&&&&&&& &-1&&&&&&& &&&&&&&& &-1&&&&&&& &&&&&&&&6 &&&&&&&& \\
111000 &&&&-1&&-1&-1& &&&&&&&&-1 &&&&&&&&-1 &&&&&&&& &&&&&&&&-1 &&&&&&&& &&&&&&&& &6&&&&&&& \\
111001 &&&-1&&-1&&&-1 &&&&&&&-1& &&&&&&&-1& &&&&&&&& &&&&&&&-1& &&&&&&&& &&&&&&&& &&6&&&&&& \\
111010 &&-1&&&-1&&&-1 &&&&&&-1&& &&&&&&-1&& &&&&&&&& &&&&&&-1&& &&&&&&&& &&&&&&&& &&&6&&&&& \\
111011 &-1&&&&&-1&-1& &&&&&-1&&& &&&&&-1&&& &&&&&&&& &&&&&-1&&& &&&&&&&& &&&&&&&& &&&&6&&&& \\
111100 &&-1&-1&&&&&-1 &&&&-1&&&& &&&&-1&&&& &&&&&&&& &&&&-1&&&& &&&&&&&& &&&&&&&& &&&&&6&&& \\
111101 &-1&&&-1&&&-1& &&&-1&&&&& &&&-1&&&&& &&&&&&&& &&&-1&&&&& &&&&&&&& &&&&&&&& &&&&&&6&& \\
111110 &-1&&&-1&&-1&& &&-1&&&&&& &&-1&&&&&& &&&&&&&& &&-1&&&&&& &&&&&&&& &&&&&&&& &&&&&&&6& \\
111111 &&-1&-1&&-1&&& &-1&&&&&&& &-1&&&&&&& &&&&&&&& &-1&&&&&&& &&&&&&&& &&&&&&&& &&&&&&&&6
\end{array}$$

6.4 Eigen Analysis Flipping bits Independently

A common mutation scheme in genetic algorithms (GAs) is not to flip a given number of bits but instead to flip each bit independently, albeit at low probability. Often the probability is set to 1/length of the bit string. Thus the mean number of bits flips is one. The actual number of flips follows a Binomial distribution but for long bit strings (i.e. large l) this can be approximated by a Poisson distribution with mean $m = 1.0$, $P(n) = m^n e^{-m}/n! = 0.37/n!$. The first few probabilities are $P(0) = 0.37$, $P(1) = 0.37$, $P(2) = 0.18$, $P(3) = 0.06$. The probabilities fall rapidly with larger changes. A complete graph Laplacian can be constructed by adding (weighted by $P(n)$) together the graph Laplacians for each value of n.

Since each each Walsh basis vector $\psi_j(x)$ is an eigenvector of all of the matrices being added together, $\psi_j(x)$ is also an eigenvector of the matrix for normal GA mutation. The eigenvalue corresponding to the j^{th} Walsh basis vector for a mutation only GA's Laplacian is

$$\lambda_j = \sum_{i=1}^{l} \frac{m^i e^{-m}}{i!} \lambda_{j,i,l} = \frac{1}{e} \sum_{i=1}^{l} \frac{\lambda_{j,i,l}}{i!}$$

(The right hand side applies where $m = 1.0$). $\lambda_{j,i,l}$ is the eigenvalue of the j^{th} Walsh basis vector when applied to the Laplacian for flipping exactly i bits in strings of length l. We set $\lambda_{j,0,l}$ to zero, since it corresponds to flipping zero bits, i.e. making no change. m is the mean number of bits flipped and λ_j is the eigenvalue of the j^{th} Walsh basis vector when applied to the Laplacian for flipping bits independently. The right most column of Table 3 gives λ_j for a mutation only GA which flips on average one bit per child.

Notice that if a fitness function is a Walsh basis functions (or a simple multiple of any) then a mutation only GA using it will have an elementary landscape. Stadler [17] also considers crossover.

Table 2: Example recursion. Each n bit Laplacian is followed by the two n-1 bit Laplacians from which it is composed and from which its eigenvectors and eigenvalues can be calculated using Equation 6 or 7. Directly below each Laplacian are its eigenvalues (which are subscripted by their multiplicities). The recursion is halted when the eigen analyse is known. I.e. for the Hamming cube ($n = 1$) or when every bit in the string is flipped ($n = l$).

$\Delta_{6,7}$					
$0_2 4_{42} 8_{70} 12_{14}$					
$\Delta_{6,6}$	$\Delta_{5,6}$				
$0_{32} 2_{32}$	$0_1 2_6 4_{15} 6_{20} 8_{15} 10_6 12_1$				
	$\Delta_{5,5}$	$\Delta_{4,5}$			
	$0_{16} 2_{16}$	$0_2 4_{20} 8_{10}$			
		$\Delta_{4,4}$	$\Delta_{3,4}$		
		$0_8 2_8$	$0_1 2_4 4_6 6_4 8_1$		
			$\Delta_{3,3}$	$\Delta_{2,3}$	
			$0_4 2_4$	$0_2 4_6$	
				$\Delta_{2,2}$	$\Delta_{1,2}$
				$0_2 2_2$	$0_1 2_2 4_1$

6.5 Largest Eigenvalue of n-bit Flip Graph Laplacian

We generalise Section 5.7, for flipping two bits, to flipping n bits. Again the smallest eigenvalue is always zero (with multiplicity one). The smallest non-zero eigenvalue corresponds to the lowest Walsh function for the Hamming cube, which is 2. As before, we can 1) establish an upper bound on the largest eigenvalue and 2) give a stochastic estimate for large l.

6.5.1 *Upper Bound on Largest Eigenvalue of n-bit Flip Graph Laplacian*

Following, Section 5.7.1 starting from Equation 7 and using $\lambda_{n,l} \geq 0$

$$\begin{aligned}
\lambda_{n,l} &= 2C^{l-1}_{n-1} + \lambda_{n,l-1} - \lambda_{n-1,l-1} \\
\lambda_{n,l} &\leq 2C^{l-1}_{n-1} + \lambda_{n,l-1} \\
&\leq 2C^{l-1}_{n-1} + 2C^{l-2}_{n-1} + \lambda_{n,l-2} \\
&\leq \lambda_{n,n} + \sum_{i=n}^{l-1} 2C^{i}_{n-1} \\
&= 2 + 2\sum_{i=n}^{l-1} C^{i}_{n-1} \\
&= 2 + 2C^{l-1}_{n-1} \sum_{i=n}^{l-1} \frac{C^{i}_{n-1}}{C^{l-1}_{n-1}} \\
&< 2 + 2C^{l-1}_{n-1} \sum_{i=0}^{\infty} \left(\frac{C^{l-2}_{n-1}}{C^{l-1}_{n-1}} \right)^{i} \\
&= 2 + 2C^{l-1}_{n-1} \frac{1}{1 - \frac{C^{l-2}_{n-1}}{C^{l-1}_{n-1}}}
\end{aligned}$$

Table 3: Eigenvalues of the graph Laplacian of mutation operators which flip exactly 1, 2, 3, 4 or 5 bits, when acting on strings of 5 bits. λ_j are the eigenvalues for elementary landscapes of a mutation only GA with $p_m = 1/5$.

	$\Delta_{1,5}$	$\Delta_{2,5}$	$\Delta_{3,5}$	$\Delta_{4,5}$	$\Delta_{5,5}$	λ_j
ψ_{00000}	0	0	0	0	0	0.000
ψ_{00001}	2	8	12	8	2	3.124
ψ_{00010}	2	8	12	8	2	3.124
ψ_{00011}	4	12	12	4	0	4.736
ψ_{00100}	2	8	12	8	2	3.124
ψ_{00101}	4	12	12	4	0	4.736
ψ_{00110}	4	12	12	4	0	4.736
ψ_{00111}	6	12	8	4	2	5.351
ψ_{01000}	2	8	12	8	2	3.124
ψ_{01001}	4	12	12	4	0	4.736
ψ_{01010}	4	12	12	4	0	4.736
ψ_{01011}	6	12	8	4	2	5.351
ψ_{01100}	4	12	12	4	0	4.736
ψ_{01101}	6	12	8	4	2	5.351
ψ_{01110}	6	12	8	4	2	5.351
ψ_{01111}	8	8	8	8	0	5.376
ψ_{10000}	2	8	12	8	2	3.124
ψ_{10001}	4	12	12	4	0	4.736
ψ_{10010}	4	12	12	4	0	4.736
ψ_{10011}	6	12	8	4	2	5.351
ψ_{10100}	4	12	12	4	0	4.736
ψ_{10101}	6	12	8	4	2	5.351
ψ_{10110}	6	12	8	4	2	5.351
ψ_{10111}	8	8	8	8	0	5.376
ψ_{11000}	4	12	12	4	0	4.736
ψ_{11001}	6	12	8	4	2	5.351
ψ_{11010}	6	12	8	4	2	5.351
ψ_{11011}	8	8	8	8	0	5.376
ψ_{11100}	6	12	8	4	2	5.351
ψ_{11101}	8	8	8	8	0	5.376
ψ_{11110}	8	8	8	8	0	5.376
ψ_{11111}	10	0	20	0	2	5.121
Largest	10	12	20	8	2	5.376
Upper bound		34	26	12.667		
Aprx. bound	5	10	10	5	1	
No. distinct	6	3	4	3	2	6

The above bound on the largest eigenvalue of the n-bit flip's Laplacian is calculated for various n and strings of length 5 at the end of Table 3.

6.5.2 *Long Bit String Estimate of the Largest Eigenvalue of the n-bit Flip Graph Laplacian*

Here we generalise the argument given in Section 5.7.2 for two bits. Again, for any unit vector u the length of Δu will be no bigger than the largest eigenvalue. Choose a random direction. I.e. let v be a vector of 2^l components each of which is either +1 or -1, chosen uniformly at random. $|v|^2 = 2^l$ hence $|v| = 2^{l/2}$. Multiply by the Laplacian.

The first element of Δv is typical of them all.

$$\Delta v(1) = \pm C^l_n \qquad \underbrace{\pm 1 \pm \cdots\cdots\cdots \pm 1}_{C^l_n \text{ terms with random signs}}$$

Since the elements of v were randomly chosen, each of the elements of Δv are independent and identically distributed, and therefore $|\Delta v(i)|^2$ are also i.i.d. Thus the expected value of $|\Delta v|^2$ is the number of components in Δv (2^l) multiplied by the expected value of any of them. E.g. the first, $|\Delta v(1)|^2$. The expected length of $\Delta v(1)$ is approximately C_n^l. (The random signs ensure on average the following terms come to near zero and for large l the sum is dominated by the largest term.) Thus the expected value of $|\Delta v|^2$ is $2^l|C_n^l|^2$ and so the expected length $|\Delta v|$ is $2^{l/2}C_n^l$. Taking the ratio $\frac{|\Delta v|}{|v|}$ gives C_n^l as an upper bound on all the eigenvalues. For long strings this will become a tight bound on the largest eigenvalue.

C_n^l is of course the number of strings which n-bit mutation can reach in one step, i.e. the degree. So our roughness measure $\lambda/d \lesssim 1$. The bound is calculated for rather short strings (length 5) at the end of Table 3.

6.6 Number of Distinct Eigenvalues

Referring back to Section 5.9. In the simple case, the string length l is an exact multiple of the number of bits to be flipped n and so the minimum number of steps to reach any point in the search space (the graph diameter) is simply l/n and hence the Laplacian has $l/n+1$ distinct eigenvalues [14, page 598]. The last line of Table 3 gives some examples.

When l/n is not an integer, calculating the graph's diameter is more complicated. For example, not every point may be accessible. I.e. the graph may fall into separate homogeneous graphs. In which case, we need only calculate the diameter of one of them. This may be slightly smaller than l/n. Secondly, the bit flips and string lengths may not align well, so even the minimum path involves repeated flips of one or more bits. In which case the diameter will be slightly more than l/n. Nevertheless, even with these complications, the number of distinct eigenvalues when flipping n multiple bits will be near $l/n + 1$.

7. COMPARISON WITH REAL GP

7.1 NIH Non-Elementary Landscape

To verify GP does behave similarly to our model, we first ran GP to show it treats EQ-parity as a needle in a haystack problem and to investigate the distribution of jump sizes in a real GP. In the first group of genetic programming runs, GP was run with Koza's 16-even parity fitness function and only EQ in the function set (details given in Table 4).

TinyGP uses the "grow" method [13] to create the initial random programs. Since there are four times as many terminals as functions, despite a large depth limit (8), the grow method produces populations that consist mostly of programs that are too small (mean 17) to solve the 16-EQ parity problem (minimum solution size 31). Note since a needle in a haystack landscape does not provide fitness guidance, the population evolves as though there was no fitness, hence there is no bloat and, excluding drift, on average programs do not change size [9]. Therefore there remains a substantial part of the population which is simply too small to solve the problem. This increases the search time. Also TinyGP does not ensure children are different from their parents. This also increases the search time. And so a large number of programs need to be run before finding a solution. The first hitting time is 2 400 000 (mean of 10 runs, standard deviation 2 200 000). Also the distribution shows signs of being geometric as expected of a needle problem. This and two other experiments are summarised in Table 5.

Table 4: TinyGP Parameters for 16 even parity

Function:	EQ
Terminals:	$D_0, \ldots D_{15}$
Fitness:	Number of correct answers on all 2^{16} test cases. However (see Section 3) only fitness 32768 and 65536 are possible.
Selection:	Steady state. 2 members tournaments.
Population:	65 536
Initial pop:	grow (max depth 8),
Parameters:	80% subtree crossover, 20% point mutation (p_m 0.05). Crossover and mutation points are chosen uniformly (i.e. without a function bias [8]) No size limit.
Termination:	100 generations

Table 5: Summary of Experiments

Problem	degree	eigenvalue	λ/d	hitting time
Parity	120	136*	1.1333*	2 400 000
4 Trap-like	120	84	0.7	18 000
1 Max	120	2	0.0167	1 020

*Parity is not elementary. We give the weighted average of all λ.

Although the programs are rather smaller than assumed in Section 4.3, if we exclude mutations which make no difference (distance 0) the distribution of jump sizes for TinyGP mutation (Figure 4 dashed line) is somewhat similar to that predicted in Section 4.3. That is, most mutations *moves* cause the child to be exactly two bits different in our search space. The number of 4, 6, 8 etc. moves falls rapidly. (As an anti-bloat mechanism TinyGP uses point mutation with a fixed probability of mutation per program element [13]. Thus not only can zero elements be changed but also a mutant child may differ in multiple places from its mother.) Notice, as expected, mutation only causes jumps by multiples of 2.

Figure 4 also shows the distribution of jump sizes caused by crossover. Note since subtree crossover can change program sizes, it can make arbitrary jump sizes (including odd sized jumps). However the most popular (mode) jump size is two, as assumed for the mutation only model presented in Section 4.3,

Most crossovers take place between trees which were themselves created by crossovers. (Point mutation does not change tree size or shape and so a tree's size and shape are determined by crossover, even if it also undergoes point mutation.) It may be these repeated crossovers that gives the distribution (excluding 0 and 1) its pleasing Zipf like tail. (Falling by about 60% per unit increase in step size.) Note, although the distribution of tree sizes would be expected to rapidly converge to a Lagrange distribution [3], the distribution in Figure 4 refers to jumps in our bit orientated semantic search space. The mapping between it and that of the GP binary trees is not straightforward and we have not attempted to prove a mathematical relationship between random crossover of Lagrange distributed trees and the jumps in our space to accompany the experimental data given in Figure 4. Figure 4 shows there is some truth in our simplification that all moves are of size two. However it shows the full situation is more complex.

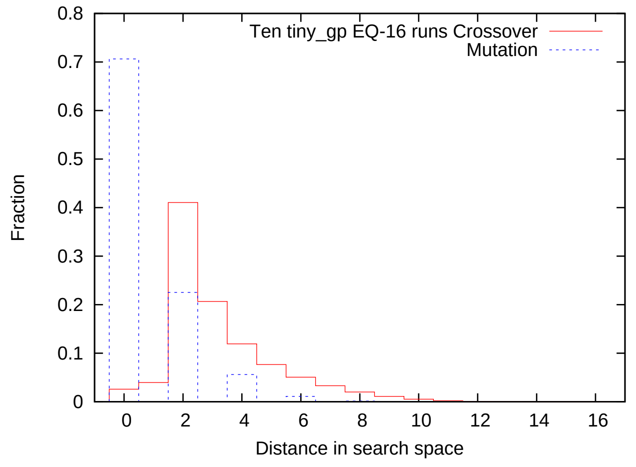

Figure 4: Distribution of jumps in the search space sizes caused by crossover and mutation in ten TinyGP 16-EQ parity runs.

7.2 1 Max Elementary GP Landscape

A further ten runs were made where the fitness function is given by the Hamming distance from the optimum in our bit-orientated semantic landscape. (Note in GP terms this is cheating, since we allow the fitness function to look inside the program, rather than simply being based on what the program does when it is run.) The problem is now like ones max and so considerably easier than the parity problem used in the previous section. n^{th} order one-max can be shown to be an elementary landscape by either noting 1) its fitness is composed of n components in Whitley's sense [22] or 2) by noting it can be decomposed into the n first order Walsh coefficients all of which are eigenvectors of Δ with the same eigenvalue and hence any linear combination of them (such as onemax) is also an eigenvector of Δ and therefor onemax is elementary. (The zeroth Walsh coefficient essentially gives the average fitness $\overline{f}$.)

TinyGP was run with a much reduced population size (128). Except for this and the fitness function, the same parameters (cf. Table 4) were used. 85% of runs succeeded in solving EQ-16 on this more friendly landscape. The mean first hitting time was 1020 (standard deviation 430). I.e. the second fitness landscape is about 2000 times easier for GP even though their search spaces are identical.

7.3 4 Trap-like Third Order GP Landscape

A further ten runs were made where the fitness function is given by summing four trap-like functions. Each trap-like function is defined on a group of four non-overlapping bits from the total of 16 in our bit-orientated semantic landscape. Where each four bit function is created by adding together all the third order Walsh coefficients. (See Figure 5. Again in GP terms this is cheating.)

If we retain Section 4's assumption that search proceeds by flipping exactly two bits, then the new fitness landscape is elementary. To show this we start with the four bit (i.e. 16×16) Laplacian Δ_5 and note the fitness function is composed of third order Walsh functions and repeat the argument given in Section 5.6 but for this special case. We have already shown the third order Walsh functions are eigenvectors of the 16×16 Laplacian and have the same

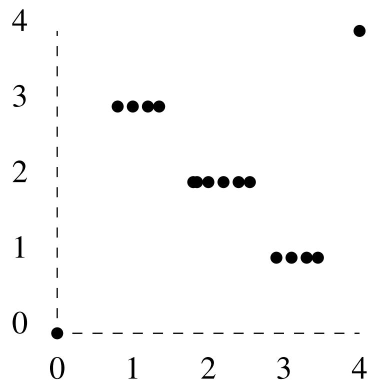

Figure 5: Schematic of the integer valued 4 bit unitation fitness function formed by adding four third order 4-bit Walsh functions. Fitness plotted vertically. Unitation horizontally. Spots indicate the fitness of 16 possible bit values. The full fitness function is formed by concatenating four of these. It has a single global optimum with fitness 16.

eigenvalue (12). The 32×32 Laplacian is formed of four 16×16 matrices as described in Equation 2. Concatenating a 16-bit Walsh function with itself and multiplying by the 32×32 Laplacian gives a vector of 32 elements, whose two 16 elements halves are both equal to a multiple of the original 16-bit Walsh function. I.e. the 32 element vector is an eigenvector of the 32×32 Laplacian. The eigenvalue is λ_{H_3} plus that of the original 16-bit Walsh function. Where λ_{H_3} is the eigenvalue of the third order Walsh function applied to the Laplacian of the Hamming cube (rather than double flip parity). $\lambda_{H_3} = 2 \times 3 = 6$. I.e. the concatenated 32 bit third order Walsh functions have eigenvalues $6+12=18$. Note this is true of all four original third order Walsh functions and therefore they all have the same eigenvalue. Therefore the 32 element vector formed by adding them together is also and eigenvector with eigenvalue 18.

We keep doubling using this procedure until we form a vector with 2^{16} elements. It will be an eigenvector of the $2^{16} \times 2^{16}$ Laplacian for the 16-EQ search space; with eigenvalue $6 \times (16-4) + 12 = 84$. Notice this vector repeats the 16 values of the third order Walsh function 4096 times, corresponding exactly to the repeat of the lower 4 bits of our trap-like fitness function. Since our landscape is symmetric we can rotate the labels by 4, 8 and 12 bits to show that the three other components of the fitness function are also eigenvectors with the same eigenvalue. Since the combined fitness function is the sum of four eigenvectors with the same eigenvalue, it too is an eigenvector of of the $2^{16} \times 2^{16}$ Laplacian (with eigenvalue 84.) Therefor our trap-like fitness landscape is elementary.

The problem is now intermediate between parity and onemax. Therefore TinyGP was run with a intermediate population size (1024). Except for this and the fitness function, the same parameters (cf. Table 4) were used. Nine out of ten runs succeeded. The mean first hitting time was 18 000 (standard deviation 9 000). I.e. this third fitness landscape is about 130 times easier for GP than the first (and about 17 times harder than the second) even though all three search spaces are identical.

8. CONCLUSIONS

We have analysed our [10] genetic programming parity fitness landscape. As well as proving it is not elementary we have calculated its mean fitness and fitness variance and its fitness distance correlation. Also we have argued that existing results on fitness autocorrelation along random walks in elementary landscapes can be greatly extended to show fitness autocorrelation falls rapidly with distance in many bit string mutation GAs.

Section 5.4 gave the elementary needle in a haystack fitness landscape.

We have given a complete eigen analysis of the search space formed by using two bit flip mutation. Showing the eigenvectors of its graph Laplacian are the same as those of the single bit flip Hamming cube (i.e. the Walsh basis functions). The eigenvalues can be rapidly calculated. They are somewhat similar but a factor of about k larger than those of the Hamming cube. Since the graph is connected the smallest eigenvalue is of course zero. The next is $2(k\text{-}1)$, the next $4(k\text{-}2)$, then $6(k\text{-}3)$ and so on. Table 1 gives the eigenvalues and their multiplicities for initial values of k. We provide two proofs (one a bound and the other a tight asymptotic limit) for the largest eigenvalue. The actual values appears to be $\lceil (k-1)(k+1)/2 \rceil$ but we have not proved this. The number of distinct eigenvalues is $k/2{+}1$ and so the separation between eigenvalues is about $2k$. For large k, multiplying the Laplacian by a unit vector pointed in almost all directions will increase its length by about the number of neighbours of each node $\frac{1}{2}k(k-1)$.

In Section 6 we generalised most of these results to landscapes where neighbours differ by $n{>}2$ bits. Indeed we gave the eigen analysis, not only for mutation which flips three or more bits, but also for normal bit string genetic algorithms (without crossover) which flip a variable number of bits independently. This showed the eigenvectors of the Hamming cube are also eigenvectors of two, three and more bit flip mutations and the mutation only GA. Hence any fitness function which is a Walsh function will form an elementary landscape with this set of widely used mutation operators. However most of these have large eigenvalues, corresponding to rough fitness landscapes. Typically an eigenvalue has a large multiplicity (i.e many eigenvectors share the same eigenvalue). Any fitness function which is a linear combination of Walsh vectors which have the same eigenvalue will also form an elementary landscape.

The number of distinct eigenvalues when flipping exactly n bits is approximately $l/n+1$ and total number of eigenvectors is 2^l. Therefore there is at least one subspace with in the region of $2^l n/l$ eigenvectors. Any linear combination of these is also an eigenvector and each of these corresponds to an elementary landscape. Restricting ourselves to coefficients 0 and 1 means this subspace alone contains at least $2^{2^l/n}$ elementary landscapes. (This is a lower bound, the total number of n-bit flip elementary landscapes is much bigger.)

We compared our simplified [10] genetic programming parity with experiment and showed it does indeed behave as a needle in a haystack. We have run GP on two elementary fitness landscapes of the same size but different fitness functions, and shown, as expected, they behave differently. I.e. elementary landscapes, with identical sizes and connectivity, can represent problems of very different difficulty. Results so far have been in keeping with the ruggedness measure (Section 5.8). However given non-universal results on other proposed indicators of problem hardness (e.g. order of non-zero Walsh coefficients) we cannot be confident of its usefulness.

Acknowledgments

I would like to thank Daniel Spielman, Gylson Thomas for fhtseq.m, Riccardo Poli for TinyGP and Gabriel Peyr for fwt.m. Also many participants at Dagstuhl Seminar 10361 and FOGA 2011 for helpful discussions, including Andrew Sutton, Francisco Chicano, Jon Rowe, Darrell Whitley, Ken De Jong, Una-May O'Reilly, Alden Wright and Nataliya Sokolovska and the FOGA 2011 anonymous reviewers.

Funded by EPSRC grants EP/D050863/2 and EP/G060525/2.

9. REFERENCES

[1] T. Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms.* Oxford University Press, New York, 1996.

[2] T. Biyikoglu, J. Leydold, and P. F. Stadler. *Laplacian Eigenvectors of Graphs: Perron-Frobenius and Faber-Krahn Type Theorems*, volume 1915 of *Lecture Notes in Mathematics.* Springer, 2007.

[3] S. Dignum and R. Poli. Generalisation of the limiting distribution of program sizes in tree-based genetic programming and analysis of its effects on bloat. In D. Thierens, H.-G. Beyer, J. Bongard, J. Branke, J. A. Clark, D. Cliff, C. B. Congdon, K. Deb, B. Doerr, T. Kovacs, S. Kumar, J. F. Miller, J. Moore, F. Neumann, M. Pelikan, R. Poli, K. Sastry, K. O. Stanley, T. Stutzle, R. A. Watson, and I. Wegener, editors, *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, volume 2, pages 1588–1595, London, 7-11 July 2007. ACM Press.

[4] B. Dimova, J. W. Barnes, and E. Popova. Arbitrary elementary landscapes & AR(1) processes. *Applied Mathematics Letters*, 18(3):287–292, 2005.

[5] B. Dimova, J. W. Barnes, E. Popova, and B. W. Colletti. Some additional properties of elementary landscapes. *Applied Mathematics Letters*, 22(2):232–235, Feb 2009.

[6] L. K. Grover. Local search and the local structure of NP-complete problems. *Operations Research Letters*, 12(4):235–243, 1992.

[7] Terry Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the 6th International Conference on Genetic Algorithms, ICGA 1995*, pages 184–192. Morgan Kaufmann, 1995.

[8] J. R. Koza. *Genetic Programming: On the Programming of Computers by Natural Selection.* MIT press, 1992.

[9] W. B. Langdon. The evolution of size in variable length representations. In *1998 IEEE International Conference on Evolutionary Computation*, pages 633–638, Anchorage, Alaska, USA, 5-9 May 1998. IEEE Press.

[10] W. B. Langdon. Scaling of program tree fitness spaces. *Evolutionary Computation*, 7(4):399–428, Winter 1999.

[11] W. B. Langdon and R. Poli. *Foundations of Genetic Programming*. Springer-Verlag, 2002.

[12] R. Poli, W. B. Langdon, and O. Holland. Extending particle swarm optimisation via genetic programming. In M. Keijzer, A. Tettamanzi, P. Collet, J. I. van Hemert, and M. Tomassini, editors, *Proceedings of the 8th European Conference on Genetic Programming*, volume 3447 of *Lecture Notes in Computer Science*, pages 291–300, Lausanne, Switzerland, 30 Mar. - 1 Apr. 2005. Springer.

[13] R. Poli, W. B. Langdon, and N. F. McPhee. *A field guide to genetic programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008. (With contributions by J. R. Koza).

[14] C. R. Reeves. Fitness landscapes. In E. K. Burke and G. Kendall, editors, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, chapter 19, pages 587–610. Springer, 2005.

[15] C. R. Reeves and J. E. Rowe. *Genetic Algorithms–Principles and Perspectives: A Guide to GA Theory*. Kluwer Academic Publishers, 2003.

[16] F. Rothlauf. *Representations for Genetic and Evolutionary Algorithms*. Physica-Verlag, 2002.

[17] P. F. Stadler, R. Seitz, and G. P. Wagner. Population dependent Fourier decomposition of fitness landscapes over recombination spaces: Evolvability of complex characters. *Bulletin of Mathematical Biology*, 62:399–428, 2000.

[18] P. F. Stadler. Landscapes and their correlation functions. Technical Report 95-07-067, Santa Fe Institute, USA, 1995.

[19] P. F. Stadler and G. P. Wagner. Algebraic theory of recombination spaces. *Evolutionary Computation*, 5(3):241–275, 1997.

[20] A. M. Sutton, A. E. Howe, and L. D. Whitley. Estimating Bounds on Expected Plateau Size in MAXSAT Problems. In T. Stützle, M. Birattari, and H. H. Hoos, editors, *Second International Workshop, Engineering Stochastic Local Search Algorithms. SLS 2009*, volume 5752 of *Lecture Notes in Computer Science*, pages 31–45, Brussels, 3-4 September 2009. Springer.

[21] A. M. Sutton, A. E. Howe, and L. D. Whitley. A theoretical analysis of the k-satisfiability search space. In T. Stützle, M. Birattari, and H. H. Hoos, editors, *Second International Workshop, Engineering Stochastic Local Search Algorithms. SLS 2009*, volume 5752 of *Lecture Notes in Computer Science*, pages 46–60, Brussels, 3-4 September 2009. Springer.

[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 *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 365–372, Montreal, 2009. ACM.

[23] V. K. Vassilev, T. C. Fogarty, and J. F. Miller. Information characteristics and the structure of landscapes. *Evolutionary Computation*, 8(1):31–60, Spring 2000.

[24] V. K. Vassilev, T. C. Fogarty, and J. F. Miller. Smoothness, ruggedness and neutrality of fitness landscapes: from theory to application. In A. Ghosh and S. Tsutsui, editors, *Advances in evolutionary computing: theory and applications*, pages 3–44. Springer-Verlag New York, Inc., 2003.

[25] D. Whitley, D. Hains, and A. Howe. Tunneling between optima: partition crossover for the traveling salesman problem. In *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 915–922, Montreal, 2009. ACM.

[26] D. Whitley, A. M. Sutton, and A. E. Howe. Understanding elementary landscapes. In M. Keijzer, G. Antoniol, C. B. Congdon, K. Deb, B. Doerr, N. Hansen, J. H. Holmes, G. S. Hornby, D. Howard, J. Kennedy, S. Kumar, F. G. Lobo, J. F. Miller, J. Moore, F. Neumann, M. Pelikan, J. Pollack, K. Sastry, K. Stanley, A. Stoica, E.-G. Talbi, and I. Wegener, editors, *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 585–592, Atlanta, GA, USA, 12-16 July 2008. ACM.

[27] L. D. Whitley and A. M. Sutton. Elementary landscape analysis. In *GECCO '09: Proceedings of the 11th annual conference companion on Genetic and evolutionary computation conference*, pages 3227–3236, Montreal, 8-12 July 2009. ACM. Tutorial.

[28] L. D. Whitley and A. M. Sutton. Partial neighborhoods of elementary landscapes. In G. Raidl, F. Rothlauf, G. Squillero, R. Drechsler, T. Stuetzle, M. Birattari, C. B. Congdon, M. Middendorf, C. Blum, C. Cotta, P. Bosman, J. Grahl, J. Knowles, D. Corne, H.-G. Beyer, K. Stanley, J. F. Miller, J. van Hemert, T. Lenaerts, M. Ebner, J. Bacardit, M. O'Neill, M. Di Penta, B. Doerr, T. Jansen, R. Poli, and E. Alba, editors, *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 381–388, Montreal, 8-12 July 2009. ACM.

[29] Tina Yu and J. F. Miller. Through the interaction of neutral and adaptive mutations, evolutionary search finds a way. *Artificial Life*, 12(4):525–551, Fall 2006.