Visualisation of Hierarchical Cost Surfaces for Evolutionary Computing

Janet Wiles[+*] and Bradley Tonkes[*]
[+]School of Psychology
[*]School of Information Technology and Electrical Engineering
University of Queensland
Queensland, 4072
Australia
{j.wiles,btonkes}@itee.uq.edu.au

Abstract - In this paper we present a technique for visualising cost surfaces that are relevant to work in evolutionary computation, particularly genetic algorithms. The focus is on moderate-dimensional, binary cost surfaces (i.e., functions defined over $\{0,1\}^n$, for $n \leq 16$) that have a hierarchical, modular structure. The visualisation approach involves an unfolding of the hyperspace into a two-dimensional graph, whose layout represents the topology of the space using a recursive relationship, and whose shading defines the shape of the cost surface defined on the space. Using this technique we present a case-study exploration of the hierarchical-if-and-only-if (H-IFF) function. The visualisation approach provides an insight into the properties of this function, particularly in regards to the size and shape of the basins of attraction around each of the local optima.

I. Introduction

A variety of techniques exist for studying the structure and properties of cost surfaces (fitness landscapes). Such approaches can provide insights for selecting appropriate search processes. The No Free Lunch theorem asserts the equality of all search processes when averaged over all cost surfaces [1]. However, it is arguable that not all cost surfaces are equally likely, so that some optimisation processes will be more effective than others *in practice* [2]. For efficient/effective optimisation, an algorithm (or its parameters) should be tailored to the features of the cost surface.

For complex families of cost surfaces, it is often a non-trivial problem to find a matching algorithm. One approach compares the empirical performance of a variety of algorithms on a suite of test functions [3]. For this approach to be informative about novel cost surfaces, there must be some principle suggesting appropriate similarities between the benchmark problem and the new problem. Knowing which metric to use to judge the similarity of two cost surfaces requires an understanding of why an algorithm performs well on a particular problem.

One barrier to understanding why an algorithm is effective is in characterising features of the cost surface. For large (benchmark) problems, where it may be too computationally expensive to compute the entire surface, one approach is to describe the surface with a set of estimated statistics, such as the number of local optima, and their relative distance. Such explorations of the cost surface have been used for describing NK landscapes [4].

An alternative approach to understanding cost surfaces, applicable to smaller benchmark problems, is the use of visualisation techniques. Visualising cost surfaces for problems of more than a few dimensions is a difficult task for the human perceptual system. Unfortunately, as is well known, surfaces in 2- or 3-D, which are most intuitive for humans, can be misleading with respect to properties of higher dimensional spaces. Ideally a benchmark problem should be small enough to be efficiently computed, while remaining complex enough to provide insights into scaled-up versions.

A wide variety of techniques exist for visualising multivariate data (see [5] for a classic guide). These include dimension-reduction techniques such as PCA and multivariate scaling (e.g., [6]), graph visualisation (e.g., [7]), and animation (e.g., [8]). Specialist techniques have also been developed in many domains. However, many techniques for visualising high-dimensional data are less well suited to visualisation of high-dimensional *surfaces*.

For different domains, the types of displays that are effective depend on the problems that are addressed and the simplifications that can be made without loss of information. This paper considers cost surfaces that have been proposed in the literature as being amenable to search by evolutionary computation, particularly using the crossover operator.

One of the early tenets of evolutionary computation was the "building block hypothesis" [9]. This hypothesis proposes that a characteristic of many real-world problems is that solutions may be constructed from progressively higher-level combinations of building blocks. These blocks must be easy to identify (once discovered) and easily recombined into a wide variety of structures. Holland [10] reasons that, "once a computer scientist starts thinking about building blocks as a source of in-

novation, the next obvious step is to look for algorithms that can discover and exploit building blocks" (p374).

Holland argues that many problems exhibit a hierarchically structured cost surface, and argues that optimisation algorithms should behave accordingly, claiming that genetic algorithms do so because of the crossover operator of reproduction which allows the recombination of disparate building blocks. The performance of genetic algorithms on such hierarchically structured cost surfaces has been studied using a variety of test functions including the Royal Road problems [11], [12]. The original Royal Road problem was proposed to study recombination using crossover, but in the search for clarity, it had only one fitness peak. It was discovered that hill-climbing strategies worked very well, better in fact than crossover (see [13] for a good summary of the reasons).

Goldberg, Deb and Korb [14] argued that demonstrating the power of crossover requires the use of deceptive functions in which the fitness landscape includes local optima that interfere with hill-climbing techniques. More recently, Watson and Pollack [15] reviewed the tasks that have been used to study hierarchical structure across a variety of studies from the literature. They concluded that many of the tasks used to investigate such problems do not have the requisite structure to fully investigate the power of evolutionary algorithms using crossover. Their own example, H-IFF (hierarchical if-and-only-if; [16]) has the interesting property of symmetry around diametrically opposed fitness peaks, with many suboptimal peaks and consequently many local optima. The H-IFF function is designed to exhibit *hierarchical* modularity, particularly *recursive* modularity (i.e., the same relationship holds between layers of the hierarchy).

This paper considers a visualisation technique for cost surfaces that are of particular relevance to genetic algorithms. The technique is applicable to cost-surfaces that are defined over moderate-dimensional binary spaces ($\{0,1\}^n$ for $n \leq 16$) and which are hierarchically structured. This visualisation tool exploits the fact that symmetric patterns in low dimensions often indicate recursive structure in higher dimensions. Although humans do not readily visualise high-dimensional structures, we do readily perceive symmetries.

The features of these functions that we would most like to understand concern

- the ruggedness of each surface, based on the number and distribution of local optima
- the size and shape of basins of attraction around each local optimum
- the length of random adaptive walks.

It is possible to use statistical techniques to estimate the properties of such features for some problems (see [4] for application to NK problems), but they require detailed mathematical insight and lack the efficacy of direct visualisation.

The visualisation technique is aimed at providing a lucid description of a cost surface, allowing researchers to understand how the properties of the cost surface interact with the properties of the evolutionary strategy (or other optimisation technique). The visualisation approach is described in Section II, and in Section III we perform a case-study analysis of the H-IFF function [16] using the visualisation tool.

II. Methods for representing cost surfaces: Hypercubes and hypergraphs

Hypergraphs are a way of representing all points in a boolean space recursively on a two dimensional display. They are particularly applicable to recursive modular functions. In this section we show how the corners of the $n-$dimensional hypercube (i.e., the points in the space) are mapped onto the two dimensional graph, and discuss some of the insights that can be gained.

Consider an $n-$dimensional binary space, $\{0,1\}^n$. Each point in this space is an $n-$dimensional binary vector, $V = (x_0, x_1, \ldots, x_{n-1})$. The set of all possible points is the set of all possible bit-strings, $000\ldots0$ to $111\ldots1$. These strings can be represented as the corners of a hypercube. Higher-order hypercubes can be recursively extrapolated from lower-order ones. A three-dimensional space has eight points which can be represented as the corners of a cube. The cube, and its unfolded hypergraph, are shown in Fig. 1. In the general case, an $n-$dimensional space has 2^n points. The corresponding hypergraph is a display of $2^{\lfloor n/2 \rfloor} \times 2^{\lceil n/2 \rceil}$ boxes, with each box representing one corner of the hypercube. A sketch of the recursive structure for $n = 8$ is shown in Fig. 2.

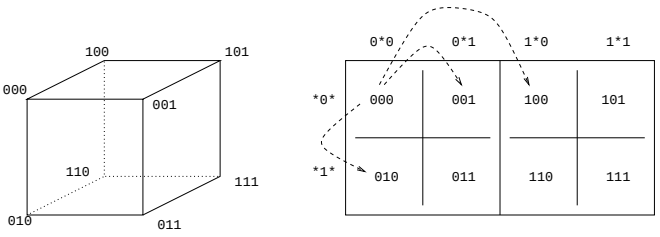


Fig. 1. Hypercube and hypergraph. This figure demonstrates the relationship between the cube (left) and its graph form (right), showing the position in the graph to which each point in the cube is mapped. Rows and columns of the graph are labelled by their hyperplane templates. Dotted arrows on the graph show the immediate neighbours of 000.

Formally, using the recursive layout described above, for each $n-$bit string, $B = b_{n-1}b_{n-2}\ldots b_0$, the cartesian co-ordinates (in binary notation) are given by $x = b_{n-2}b_{n-4}\ldots b_0$ and $y = b_{n-1}b_{n-3}\ldots b_1$. Thus, the lower order bits define the fine structure of the hypergraph and the higher order ones define the gross structure. The string of all zeroes $(000\ldots0)$ maps to the top left corner, and the vector of all ones $(111\ldots1)$ maps to the bottom right corner. The structure of the display can be tuned to the problem under investigation. For example, to keep adjacent dimensions together an alternative layout is to represent the lower order bits on the $x-$axis, and the higher order bits on the $y-$axis, corresponding to an alternative unfolding of the hypercube.

Each point in the space has n immediate neighbours, which are the bit strings at a Hamming distance of one. In the graph, these neighbours are located at points $1, 2, 4, \ldots, 2^{n/4}$ positions away in the same row and column. It is often helpful to overlay grid-lines on the graph which vary in thickness and highlight the recursive symmetries in the graph (see Fig. 2).

The regularity of the hypercube connection structure makes it possible to view the hypergraph without explicitly representing the neighbourhood topology. Since the hypergraph is a recursive unfolding of the hypercube, rather than a low-dimensional projection, each corner of the hypercube has a unique position on the graph and no information is lost in the process. Thus, if required, the positions of all connections can be inferred from the position of each string in the hypergraph.

The hypercube has been unfolded onto the hypergraph using translations for each dimension. Hence, recursive symmetries characterise the display. Consider an 8-dimensional space. There are $2^8 = 256$ strings in this space. In GA terms, each allele (0 or 1) in a string defines a hyperplane that divides the space in half. For example, $*******1$ corresponds to a set of rows and $*0******$ corresponds to a set of columns. Combinations of alleles such as $*0*****1$ (also known as *clusters*) correspond to intersections of the separate hyperplanes.

Given the basic hypergraph layout, we can assign a cost to each point, and plot each point as a coloured box on the hypergraph (see next section for examples). The lower the cost of a particular point (i.e., the more optimal it is) the lighter the colour it is shaded. Thus, points of maximal cost are shaded white, and points of minimal cost are shaded black. The set of points in the hypergraph shows the entire cost surface.

The hypergraph visualisation technique has been implemented as a Java tool which allows several properties of interest to be explored. These include

- the number and distribution of local and global optima (peaks in the landscape)
- points of low fitness, or valleys
- paths from a point to those fitness peaks that can be reached via fitter neighbours (i.e., adaptive walks)
- steepest ascent paths from a point
- basins of attraction for different peaks (the set of points that can climb to that peak)
- (inverse) basins of potential (the set of points that can be reached via an adaptive walk from a point)
- the extent and shape of neutral layers (sets of neighbouring points of equal fitness that provide agents with starting points for exploring different peaks; H-IFF does not exhibit this characteristic).

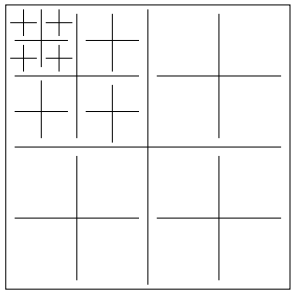


Fig. 2. Hypergraph layout for an 8-dimensional space. The 8D hypercube is recursively unfolded to show all 256 strings, with 00000000 in the top left corner, and 11111111 in the bottom right corner.

III. H-IFF: A Case Study

A. H-IFF Explained

A variety of modular tasks have been proposed to study the conditions under which GAs outperform comparable search techniques. Hierarchical-If-and-only-If (*H-IFF*) [16] is one such function that is hierarchical, modular, is not searchable by mutation, but is amenable to search by crossover. Its defining characteristics are two fitness peaks at opposing corners of the search space. Combinations of the sub-components that comprise each level of the competing hierarchies cause many sub-optimal peaks and consequently many local optima.

H-IFF [16] is defined on bit-strings of length 2^n. The fitness value of a particular string is defined in terms of hierarchical 'building blocks' which are sub-strings of the main bit-string. The building block at the highest level of the hierarchy is defined over the entire bit-string (i.e., all 2^n bits). Each building block is recursively divided into two equally-sized blocks, except for blocks of size one, which cannot be divided. For a building block to be correctly set, it must consist of either all 1s or all 0s. The value of a correctly set building block of size n is 2^n *plus* the sum of the values of its two sub-blocks (whose

values depend on the sub-sub-blocks). Thus, the overall value of a bit-string of length 2^n is the sum of values for the building blocks of sizes $1, 2, 4, \ldots, 2^n$. The optimal bit-strings consist of either all 0s or all 1s so that they are rewarded for building blocks of every size. The evaluation of the H-IFF function is more easily understood by way of example, shown for an 8-bit string in Fig. 3.

0	**0**	**1**	**0**	**1**	**1**	**1**	**1**	Value
1	1	1	1	1	1	1	1	8
2		—		2		2		6
—				4				4
—								0

Fig. 3. Evaluation of H-IFF for the bit-string 00101111 showing hierarchical decomposition. For this bit-string, H-IFF evaluates to $8 + 6 + 4 + 0 = 18$. Note that the maximum obtainable value for each level of the hierarchy is 8, so the maximum value for H-IFF on bit-strings of length 8 is 32.

For H-IFF defined on $l = 2^n$ bits there will be $n + 1$ levels of building blocks. Within these levels there will be l/k building blocks of size k, each of which has value k. The optimal bit-strings of length l therefore have value $l(n+1)$. The minimum value for H-IFF on bit-strings of length is l. Such a bit-string contains all building blocks of size 1 but no higher-level blocks.

The major difference between H-IFF and the more well known Royal Road function is that RR has a single optimal bit-string (all 1s) and significantly, *no local optima* other than the global optimum (although there are local plateaus). By comparison, H-IFF has two optimal bit-strings and, for bit-strings of length $l = 2^n$, there are $2^{l/2}$ local optima.

B. Visualising H-IFF

Our first example of the hypergraph visualisation approach takes a very simple instantiation of H-IFF, defined over only four dimensions (see Fig. 4). The symmetry of the display instantly reveals the symmetry of the underlying function.

Examining a hypergraph for a larger space (see Fig. 5) shows the generality of the structures observed in Fig. 4. The visualisation tool allows some aspects of the cost surface to be directly displayed.

Number of local optima. The local optima can be highlighted, and doing so reveals that they always fall along the diagonal (as observed in the four-dimensional case). Thus, the number of local optima is the square root of the size of the space, $2^{n/2}$. Furthermore, there are an equal number of local pessima (points that have

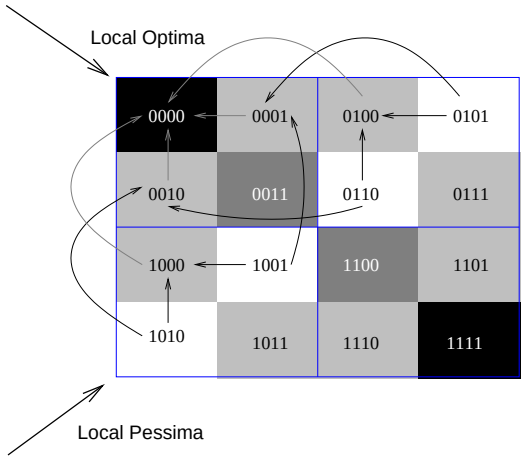


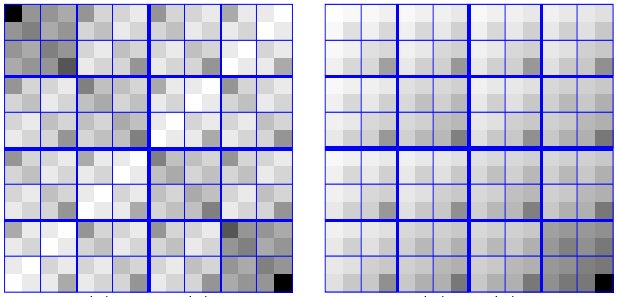
Fig. 4. Hypergraph representation for H-IFF defined on four dimensions. Darker shaded areas correspond to points of less cost (i.e., more optimal). Interactions with the visualisation tool reveal that the local optima all fall along the top-left/bottom-right diagonal, and that the points of least fitness (the pessima) fall along the other diagonal. Also shown in this figure are all of the paths of increasing fitness that lead to one of the global optima (0000). The set of points through which these paths travel form the basin of attraction for 0000. The basin of attraction for the other global optimum can be easily derived through the symmetry of the graph.

no worse neighbours) which fall along the other diagonal, all of which are *global* pessima.

Size and shape of basins of attraction. Some insights can be drawn about the size and shape of the basins of attraction for the local optima, shown in Fig. 6. The most obvious observation is that that basin of attraction for the global optimum is a Sierpinski triangle (made more obvious by the manner in which points are highlighted in Fig. 6). The recursive structure shows that the size of each basin is given by $3^{n/2}$. Thus, the basins scale in proportion to *(the number of points in the Sierpinsky triangle) / (the size of the space)*, or $3^{n/2}/2^n = (3/4)^{n/2}$.

Not only does the basin of attraction form a Sierpinski triangle for the global optima, the basins of attraction for local optima are also variants of the Sierpinski triangle (see Fig. 6b). Thus, a cursory visual inspection using the hypergraph tool reveals that the basins of attraction for all local optima are of the same size. The size and shape of these basins suggests that random adaptive walks are as likely to terminate at the global optima as they are to terminate at a local optimum (i.e., all outcomes are equally likely). Further detailed analysis revealed that this speculation was indeed correct.

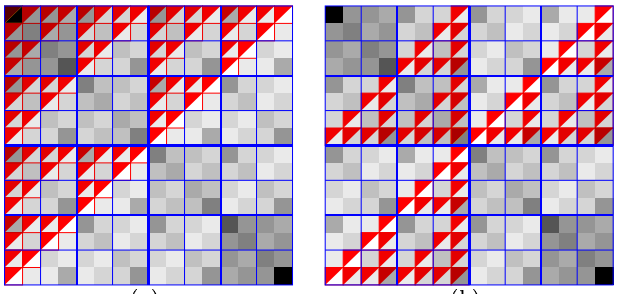
Interactions between basins of attraction. Such a landscape is very difficult to optimise using hill-



(a) H-IFF(8) (b) RR(8)

Fig. 5. (a) Eight-dimensional H-IFF. As in Fig. 4, all local optima fall along the top-left/bottom-right diagonal, and all pessima fall along the other diagonal. Compare the structure of H-IFF with that of Royal Road, shown in (b). The landscape of H-IFF is clearly more complex than that of RR which has only a single fitness peak (at 11111111, in the lower right corner) and which is, effectively, a convex bowl: the basin of attraction for the global optimum is the entire space.

climbing approaches. The only points in common between the basins of attraction for 00000000 (Fig. 6a) and 11111111 (not shown, but symmetric) are the pessima, which lie along the diagonal. In fact, the pessima belong to the basin of attraction for all local optima, or conversely, any local optimum can be reached by an adaptive walk from any pessimum. The separation of the basins at such a low level of fitness indicates a watershed in the landscape that would be difficult to traverse with hill-climbing algorithms. The H-IFF task was designed to be difficult for mutation alone, and the hypergraph makes the success of this design goal directly apparent.



(a) (b)

Fig. 6. Eight-dimensional H-IFF. Shown are (a) the basin of attraction for the global optimum, 00000000 and (b) the basin of attraction for the local optimum 00111111 (the bottom right point in the upper left quadrant). Points in the basin have been highlighted by drawing a triangle across the upper left half of the area. (In the interactive tool, the whole square is coloured. The tool was modified to allow display on greyscale media.)

IV. Evaluation

Hypergraphs provide a mechanism through which properties of high-dimensional surfaces can be explored with relative ease. Visualisation provides direct insight into landscape structure negating the need for mathematical expertise. Early unpublished experiments on the usability of hypergraphs showed that for six-dimensional hypercubes, people either mastered the display and learned to navigate very quickly — usually in time linearly proportional to the Hamming distance between points — or failed to understand the structure at all. One subject, a computer science graduate student, not only had the fastest times of any of the people tested, but also appeared to navigate between points in constant time.

The hypergraph visualisation technique is not without its shortcomings. Primary amongst its limitations is its lack of scalability. Because the hypergraph attempts to represent the entire surface (a worthy goal), the size and complexity of the display scales exponentially in the dimensionality of the cost function. It is not expected that the approach be used in exploring real-world problems, since if the cost surface can be enumerated, the function poses little practical difficulty. Even on some artificially constructed test functions, the cost surfaces for problems of 'interesting' size are too large to represent. For example, Holland's [10] hyperplane defined functions (hdf) are defined over spaces of hundreds of dimensions.

Nevertheless, some simplifications can be made that allow these larger, and more complex cost functions to be reduced to manageable sizes: much of the space in hdfs is a flat plateau of uniform (maximal) cost which can be removed by the deletion of 'uninteresting' dimensions. On these simplified versions of hierarchically defined test functions, the hypergraph display provides useful intuitions of the high-dimensional structure.

A further issue with applying the hypergraph approach to hierarchical, modular functions is the mismatch between the topology of the graph and the manner in which GAs search. The unfolding of the space into the hypergraph structure is done so that neighbouring points are laid out in a recursive fashion. That is, the layout of the hypergraph is designed to visually enhance the *Hamming distance* between points: translational adjacency in the hypergraph topology represents single-point mutation. The layout of the hypergraph is thus tuned to display how the space would be searched by a hill-climbing approach. Crossover works by combining subsequences from potentially disparate points. The hypergraph does not make it clear how a population of solutions would adapt when crossover is used. A possible solution to this dilemma might be to animate the adaptation of a pop-

ulation, but the dimension limitations, discussed above, may limit the viability of this approach.

As mentioned in the introduction, the hypergraph visualisation technique is aimed at exploring cost surfaces that have a hierarchical, modular structure defined over a binary space. To this end, the technique is successful (modulo the reservations above) in that it uses the human perceptual system's ability to perceive symmetry as a way to enhance the salient features of recursive high-dimensional structure. We have also successfully used hypergraphs for investigating non-hierarchical surfaces, particular NK functions [4]. For these surfaces, the hypergraph is not unfolded recursively, rather, neighbouring dimensions are grouped together along an axis (i.e., low-order dimensions are along the $x-$axis and high-order dimensions are along the $y-$axis).

V. Discussion and Conclusions

The H-IFF function was originally developed as a case study through which the search strategy of building-block recombination in GAs was demonstrably superior to mutation-based search [16]. The insights provided by the hypergraph visualisation make it clear why mutation-based search was so unsuccessful. Observations that the size of the basins of attraction were equally sized prompted the hypothesis that all outcomes of random-adaptive walks were equally likely. The graphs also made it clear how the number of local optima scaled with the size of the problem. From these two properties it follows that the size of the population for mutation-based search must scale with the number of local optima ($2^{n/2}$). Thus, from visualisation of the cost surface we (a) gain insight into the most desirable parameters for mutation-based search, and (b) discover that module recombination is a more appropriate approach.

Comparing the visual properties of H-IFF to those of RR (Fig. 5) reveals an obvious difference. Relating the performances of different algorithms on these two cost surfaces to features in their visual appearance provides insight into novel problems. Upon recognising that the pattern of RR represents a convex bowl it becomes easy to identify this pattern as a feature of other surfaces (e.g., smooth NK functions). The efficacy of the visual display provides insights into appropriate similarity metrics between cost surfaces (i.e., the features that are important for a particular optimisation algorithm). For example, recursive structure in high dimensions produces repeated patterns in a low dimensional unfolded representation.

The visualisation approach taken in this paper offers some benefits over the more traditional statistical and mathematical analysis of cost surfaces. One benefit is that it is more accessible to those lacking a mathematical background. Another is that it provides an exploratory tool: interesting features 'pop out' and can then be subjected to a more rigorous analysis as was the case with H-IFF, where we determined that all local optima were equally likely to be found by a random adaptive walk.

A Java-based version of the visualisation tool is available at `http://www.itee.uq.edu.au/~btonkes/hsgp.html`.

References

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

[2] S. Christensen and F. Oppacher. What can we learn from No Free Lunch? A first attempt to characterize the concept of a searchable function. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1219–1226, San Francisco, CA, 2001. Morgan Kaufmann.

[3] T. C. Belding. Potholes on the royal road. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 211–218, San Francisco, CA, 2001. Morgan Kaufmann.

[4] S. Kauffman. *The Origins of Order*. Oxford University Press, NY, 1993.

[5] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphic Press, reprint edition edition, 1992.

[6] J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, 23(9):881–889, September 1974.

[7] A. N. Pryke. *The Haiku Visualisation System*. PhD thesis, University of Birmingham, 1996.

[8] A. Buja and D. Asimov. Grand tour methods: An outline. In *Proceedings of the 18th Symposium on the Interface*, pages 63–67. American Statistical Association, 1986.

[9] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[10] J. H. Holland. Building blocks, cohort genetic algorithms, and hyperplane-defined functions. *Evolutionary Computation*, 8(4):373–391, 2000.

[11] S. Forrest and M. Mitchell. What makes a problem hard for a genetic algorithm? Some anomalous results and their explanation. *Machine Learning*, 13(2/3):285–319, 1993.

[12] M. Mitchell, J.H. Holland, and S. Forrest. When will a genetic algorithm outperform hill climbing. In *Advances in Neural Information Processing Systems*, volume 6, pages 51–58, 1994.

[13] M. Mitchell. *An introduction to genetic algorithms*. MIT Press, Cambridge, MA, 1996.

[14] D. E. Goldberg, K. Deb, and B. Korb. Messy genetic algorithms: Motivation, analysis and first results. *Complex Systems*, 3:493–530, 1989.

[15] R.A. Watson and J.B. Pollack. Hierarchically-consistent test problems for genetic algorithms. In P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, editors, *Proceedings of 1999 Congress on Evolutionary Computation*, pages 1406–1413. IEEE Press, 1999.

[16] R.A. Watson, G.S. Hornby, and J.B. Pollack. Modeling building-block interdependency. *Parallel Problem Solving from Nature, proceedings of the Fifth International Conference*, pages 97–106, 1998.

[17] M. Pelikan and D. E. Goldberg. Hierarchical problem solving by the bayesian optimization algorithm. IlliGAL Report No. 2000002, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL, 2000.