Hierarchically Consistent Test Problems for Genetic Algorithms

Richard A. Watson **Jordan B. Pollack**

Dynamical and Evolutionary Machine Organization
Department of Computer Science, Brandeis University
Waltham Massachusetts USA
{richardw, pollack}@cs.brandeis.edu

Abstract- The Building-Block Hypothesis suggests that the genetic algorithm (GA) will perform well when it is able to identify above-average-fitness low-order schemata and recombine them to produce higher-order schemata of higher fitness. We suppose that the recombinative process continues recursively, combining schemata of successively higher orders as search progresses. Historically, attempts to illustrate this intuitively straight-forward process on abstract test problems, most notably the Royal Road problems, have been somewhat perplexing. More recent building-block test problems have abandoned the multi-level hierarchical structure of the Royal Roads, and thus departed from the original recursive aspects of the hypothesis. This paper defines the concept of *hierarchical consistency*, which captures the recursive nature of problems implied by the Building-Block Hypothesis. We introduce several variants of problems that are hierarchically consistent and begin to explore aspects of problem difficulty with respect to these models.

1 Introduction

The Building-Block Hypothesis [Holland 1975, Goldberg 1989] suggests that GAs perform well when they are able to identify above-average-fitness low-order schemata and recombine them to produce higher-order schemata of higher fitness. We suppose that this process continues recursively, combining schemata of successively higher order as search progresses. More simply, the GA first finds solutions to small sub-problems and then puts these together to find solutions to bigger sub-problems, and so on to find a complete solution. The Royal Road (RR) problems [Mitchell et al 1992, Forrest & Mitchell 1993] were an early attempt to illustrate the hierarchical process of building-block discovery and recombination. However, it was apparent in the early work on RRs that there was something amiss [Forrest & Mitchell 1993b]. The functions were intended to exemplify the class of problems that required the recombinative aspects of the GA. However, non-recombinative algorithms, such as the Random Mutation Hill-Climber (RMHC), outperformed the GA on the Royal Roads. This contributed significantly to the controversy surrounding the Building-Block Hypothesis and the utility of recombination [Horn & Goldberg 1994, Mitchell et al 1995, Altenberg 1995, Jones 1995].

We argue that the reason these models fail to exemplify the benefit of recombination is because they fail to follow-through with the recursive implication of the Building-Block Hypothesis - at least, they do not follow-through consistently. This paper builds upon the work presented in [Watson et al 1998] and the hierarchically consistent test problem, H-IFF, introduced therein. We review the definition of H-IFF and define the concept of hierarchical consistency.

Following on from the work of Forrest and Mitchell [1993b], we address the question, "What makes a *hierarchically consistent* problem hard for a GA?" We expand the canonical form of H-IFF into other hierarchically consistent functions, and more general cases, that allow us to vary the difficulty of the problem.

2 Background

The primary suspect for the failure of the Royal Roads was their lack of *deception*. Deception comes in more than one variety [Horn & Goldberg 1994] but the general idea is that "a deceptive function is one in which low-order schema fitness averages favor a particular local optimum, but the global optimum is located at that optimum's complement" [Goldberg et al 1989].

The Royal Roads do not exhibit any local optima; and, with the benefit of hindsight, it is not surprising that a variant of a hill-climber could solve them. Accordingly, the original Royal Roads have largely been discarded - their purpose served - as a stepping stone on the way to better understanding. They have been replaced by building-block functions which have some degree of deception as we shall discuss. But, the Royal Roads attempted to capture a property that other building-block functions have not addressed - specifically, they included a multi-level hierarchical structure. In order to regain the hierarchical aspects of the Building-Block Hypothesis in a test problem we will re-examine the RRs and compare them to building-block functions that incorporate deception.

2.1 Hierarchical combinations

The Royal Road functions consist of a set of building-blocks - above-average-fitness schemata of short defining length [Holland 1975, Goldberg 1989]. There are eight blocks of eight bits each. Each block confers a fitness contribution, and in the second version of the Royal Road problem, R2 [Mitchell and Forrest, 1993], there are also bonus fitness contributions for particular combinations of blocks in a pairwise hierarchical fashion. Unfortunately, although this sounds like the right idea for a hierarchical building-block problem, its implementation was flawed.

The solution to each block is defined by a particular combination of bits and the number of possible combinations of 8 bits is 256. This defines the problem at the first level. The problem at the next level concerns combinations of blocks. There are eight blocks in the second level (of the R1 variant) just as there are eight bits in the first level - so far, the problem is consistent. However, since the fitness contributions of the first level identify the correct configuration of bits in a block uniquely there is no work to be done at the next level. Whereas there are 256 combinations of the bits in any block there is only one way to combine the 8 blocks since (unlike the bits) each block only has one setting. To put it another way, it is not possible to define any interdependency between blocks when the solution to each block is identified uniquely [Watson et al 1998]. That is, the blocks are *separable* - meaning that the *optimal settings* for the bits of a block are not dependent on the settings of bits in any other block.[1]

2.2 Competing schemata

Building-block problems that incorporate deception, for example the concatenated trap functions [Goldberg et al 1989], do reward more than one combination of bits within each block. Specifically, the deceptive schemata are the complement of the schemata that contain the global optima. However, they fail to follow-through with the recursive implications of the Building-Block Hypothesis. Rather than define interactions between the blocks using the two possible kinds of block (in a manner similar to that used to define the interdependency between the bits), the whole problem is simply a concatenation of many of the base deceptive functions.

2.3 Competing-schemata and hierarchy

To resolve these issues we can combine the competing schemata found in the base level of the concatenated trap functions with the hierarchical aspect of the Royal Roads. The Hierarchical-if-and-only-if (H-IFF) problem introduced by Watson et al [1998], has a hierarchical building-block structure not unlike R2. However, H-IFF defines *two* above-average-fitness configurations for each block - analogous to the competing schemata in the deceptive traps. Then fitness contributions at the next level in the hierarchy are defined via combinations of these two kinds of block. In this manner the interdependency between blocks can be defined in exactly the same manner as the interdependency between bits. H-IFF is the first example of a building-block test problem which is hierarchically consistent.

[1] This observation does not contradict the fact that the *fitness contribution* of a block may vary as a non-linear function of the settings of bits in other blocks [Whitley et al 1995].

3 Hierarchical Consistency

Before defining hierarchical consistency it will assist us to review our canonical example: Hierarchical-if-and-only-if.

3.1 Hierarchical-if-and-only-if (H-IFF)

We may view H-IFF as a problem defined over successive levels of building-blocks. Briefly, at the bottom level each non-overlapping adjacent pair of bits constitutes a block and has two solutions. Specifically, the block confers a fitness contribution if the bits are either both zero or both one. Similarly, at the second level in the hierarchy, pairs of blocks from the first level confer additional fitness if they are equal i.e. all 4 bits are zeros or all 4 are ones. Blocks of 8, 16 etc. are rewarded over subsequent levels up to the complete string. The fitness contribution for a block of one bit is 1, and the fitness contributions double at each level, keeping lock-step with the size of the block. More formally, the fitness of a string using H-IFF can be defined using the recursive function, given below. This function interprets a string as a binary tree and recursively decomposes the string into left and right halves. In this manner, a string is evaluated by summing the fitness contributions of all sub-blocks at all levels.

$$f(B)=\begin{cases} 1, & \text{if } |B|=1, \\ |B| + f(B_L) + f(B_R), & \text{if } (|B|>1) \text{ and } (\forall i\{b_i=0\} \text{ or } \forall i\{b_i=1\}), \\ f(B_L) + f(B_R), & \text{otherwise.} \end{cases}$$

where B is a block of bits, $\{b_1,b_2,...b_n\}$, $|B|$ is the size of the block=n, b_i is the ith element of B, and B_L and B_R are the left and right halves of B (i.e. $B_L=\{b_1,...b_{n/2}\}$, $B_R=\{b_{n/2+1},...b_n\}$). *n* must be an integer power of 2.

Some features of this apparently simple function should be highlighted. The structure of H-IFF is very close to the structure of R2 in some respects, but importantly, in R2 there is only one solution to each block, whereas in H-IFF there are two competing high-fitness schemata for each block at each level. Local optima in H-IFF occur when incompatible building-blocks are brought together. For example, consider "11110000"; viewed as two blocks from the previous level (i.e. size 4) both blocks are good - each contains one of the two global optima - but when these incompatible blocks are put together they create a sub-optimal string that is maximally distant from the next best strings i.e. "11111111" and "00000000".

Deception in H-IFF is more to do with incompatibility with *context* than incompatibility with a global optima as it is usually defined. Note that although local optima and global optima are distant in Hamming space they are close in recombination space [Jones 1995]. Thus H-IFF exemplifies the class of problems for which recombinative algorithms are well-suited.

Our previous work showed that H-IFF is easy for a GA to solve given that diversity in the population is maintained and genetic linkage is tight. We shall address diversity maintenance methods in the experiments of this paper. Algorithms to address poor linkage are addressed in [Watson & Pollack, 1999].

3.2 Defining hierarchical consistency

In our previous work a *hierarchically consistent* problem was described as a problem where "the nature of the problem is the same at all levels in the hierarchy." Here we state this more precisely. First, let us note that a problem can only be defined with reference to a set of variables. We usually consider the variables of GA problems to correspond directly to the genes and this is the case for the first level, level 0, in a hierarchical problem. The problem at level 0 will be to find good combinations of bits - above-average-fitness low-order schemata. However, at the next level, the problem is not to find good combinations of *bits*, but to find good combinations of *schemata* from level 0. The key to hierarchical consistency is to recognize that the variables over which a problem is defined scale-up as we ascend levels. Using this concept, we now formally define hierarchical consistency as follows:

> **Definition**: A problem is *hierarchically consistent* if for some $\boldsymbol{K}$, the problem of finding good schemata of length $\boldsymbol{L}$, given good schemata of length $\boldsymbol{L}/\boldsymbol{K}$, is of the same class for all $\boldsymbol{L}$.

For discussion of H-IFF we will permit the limitation that $\boldsymbol{L}$ is an integer power of $\boldsymbol{K}$, and naturally, $\boldsymbol{L} \leq \boldsymbol{N}$, the total size of the problem. The definition does not dictate which class of problem is involved at each level. It will likely include some kind of interdependency between variables assuming we want the problem to be non-trivial, but other properties may be varied. For example: the partitions of subproblems may be neat or overlapping; it may include linkage difficulties; and it may be real-valued or discrete. There need not even be distinct hierarchical levels since the restriction that L is a power of K can be relaxed. The definition is merely concerned with the *consistency* of the problem at different scales of resolution.

In H-IFF there are two *varieties* of solution to level-1 blocks, the 0 kind and the 1 kind, i.e. "00" and "11". This is analogous to the two varieties of bits at level-0. So, we can define the fitness contribution for level-2 blocks (size 4) using specific combinations of level-1 blocks in exactly the same way as we did with the bits level-0 - i.e. the solution for a level-2 block is that it contains two level-1 blocks of the same kind. This consistency is maintained through all levels.

R2 cannot be hierarchically consistent since there is only one variety of solution to each block (i.e. all ones) whereas there are two varieties of bits. The concatenated trap functions do, in fact, have two notable above-average-fitness schemata for each block - and, just like H-IFF, these are all-1s and all-0s. But the 0s schemata are not considered to be 'solutions' - they are of no use to the next level of building-blocks - they are merely traps incorporated to foil hill-climbers. Thus the concatenated trap functions are also not hierarchically consistent.

3.3 Constructing Hierarchically Consistent Problems

In the previous discussion it will be clear that we are interested not only in the fitness contribution of a subsolution but also in *which* solution it is. In H-IFF, for example, there are two kinds of solutions - the all-1s kind and the all-0s kind - and we cannot distinguish them by their fitness. So it is clear we cannot define the interdependency of blocks as a function of their fitness - it must be a function of their kind. Notice that although there are four possible assignments of bits for a two-bit block only two of these are of concern - i.e. the two kinds of solution. This constitutes *dimensional reduction*.

Without the insight of dimensional reduction, H-IFF is a seemingly impossible problem. Since the optimal setting for each block, at any level, including the top level, is dependent on the setting of all other blocks (i.e. they have to be the same kind to confer the optimal fitness), the optimal setting for each bit in the whole problem is dependent on the setting of every other bit. But the feature of dimensional reduction enables an algorithm to get purchase on these problems. Although every bit is dependent on every other, we need not consider all possible combinations of bits. For example, in H-IFF, at the top level we need only consider the four possible combinations of the length N/2 blocks. The Building-Block Hypothesis describes exactly this kind of dimensional reduction when it refers to combinations of schemata that supersede combinations of bits.

These considerations lead us to construct hierarchically consistent problems using two functions - one defining the fitness contributions of blocks and the other defining the kind or variety of blocks. We shall call the latter the 'transform' function since it defines which kind of block a collection of bits is transformed into for the next level in the hierarchy. The transform function defines the dimensional reduction - it transforms a block of symbols into a single symbol of the same alphabet and thus, as it is applied over successive levels, incrementally reduces the dimensionality of the problem.

For example the transform function for H-IFF is defined to reduce two symbols to one symbol as follows:

$$\boldsymbol{t}(a,b)=\begin{cases} 0, & \text{if } a=0 \text{ and } b=0, \\ 1, & \text{if } a=1 \text{ and } b=1, \\ \text{null}, & \text{otherwise.} \end{cases}$$

Notice that $\boldsymbol{t}$ defines a non-solution, null, as well as the two solutions, 0 and 1. Importantly, to allow recursion, the definition of $\boldsymbol{t}$ holds for all $\boldsymbol{a}$ and $\boldsymbol{b}$ from the ternary alphabet {0, 1, null}. The fitness contributions are then defined over this alphabet as follows:

$$\boldsymbol{f}(a)=\begin{cases} 1, & \text{if } a=1 \text{ or } a=0, \\ 0, & \text{otherwise.} \end{cases}$$

It is the job of the fitness function to direct search toward the useful schemata for the next level. Accordingly H-IFF rewards the 0 kind and the 1 kind of block but does not reward the null, or non-solution blocks. These two base functions, $\boldsymbol{f}$ and $\boldsymbol{t}$, are utilized by two corresponding recursive functions, $\boldsymbol{F}$ and $\boldsymbol{T}$, that define the fitness of a whole string and the transform of a whole string respectively. $\boldsymbol{F}$ and $\boldsymbol{T}$, defined below, decompose a string into its constituent blocks, and sum the fitness contributions from every block at every level. The fitness contributions are scaled according to their size.

$$F(B) = \begin{cases} f(B) & if\ |B| = 1, \\ |B|\ f(T(B)) + \sum_{i=1}^{k} F(B^i) & otherwise. \end{cases}$$

$$T(B) = \begin{cases} b_1 & if\ |B| = 1, \\ t(T(B^1),...,T(B^k)) & otherwise. \end{cases}$$

where ***f*** is a base function, $\boldsymbol{f}:\alpha \rightarrow \Re$, giving the fitness of a single symbol, ***t*** is a base function, $\boldsymbol{t}:\alpha^k \rightarrow \alpha$, that defines the resultant symbol from a block of k symbols, |B| is the number of symbols in B, and B^i is the i^{th} sub-block of B i.e. $\{b_{(i-1)d+1}, ..., b_{id}\}$ (d=|B|/k).

4 Hierarchically Consistent Variations

Separating the recursive structure (***F*** and ***T***) from the specifics of the problem (***f*** and ***t***) in the above manner enables us to generalize from H-IFF to other hierarchically consistent problems. This section defines alternate base functions, ***f*** and ***t***, and discusses the features they produce in the overall problem. In all cases the recursive construction functions, ***F*** and ***T***, are used unchanged together with these base functions to complete the definition of each variant problem. ***F*** and ***T***, ensure that the resultant overall problem is hierarchically consistent.

Our intent is to begin to dimensionalize the difficulty of hierarchically consistent problems and gain some understanding of what makes a problem hard for a GA - as was the original goal of the Royal Road problems. Existing measures of problem difficulty cannot be directly transferred to hierarchically consistent problems. For example, the concept of *order-k delineation* formalizes the belief that there is an upper bound on the order of interdependencies in a problem - *k*, in this concept is "the highest order deceptive nonlinearity suspected in the subject problem" [Goldberg et al 1989]. Hierarchically consistent problems cannot be delineated into separable sub-problems.

Candidates for varying difficulty in hierarchically consistent problems, addressed below, include: varying the difficulty of the base function by changing the alphabet size or the number of symbols per block; changing the fitness contributions of competing schemata; and changing the nature of the interaction of blocks from one level to the next. There are many other ways in which hierarchically consistent problems could be varied including real-valued models, and models with less strict partitioning between blocks. Here we are just beginning to explore some simple variations.

4.1 Cross-Sections

To help us obtain an intuitive feel for the variations we are about to introduce, we will employ cross-sections through the fitness landscapes they define. In particular, we choose a section from all zeros to all ones. For example, see the curve labeled 'hequal-2' in Figure 1c - (this curve, as we shall explain, is identical to that for H-IFF). The first point on this cross-section is the fitness of the all zeros string for a 64-bit problem; the second point is the fitness of a string starting with a single 1 and followed by 63 zeros; the third, 2 leading ones and the remainder zeros, and so on. Note that in the original H-IFF the two ends of this section (i.e. all ones and all zeros) are the global optima, and that the best local optima are at half ones followed by half zeros or vice versa. A point with this second-best fitness therefore appears in the middle of the section. The recursive aspect of the functions are clear in these sections; each half section is to the whole section as each quarter section is to the half section. Although illuminating, we must be cautious in the use of these sections. We are showing only 65 points out of a total 2^{64}, and only 33 local optima are evident out of a total 2^{32}.[2]

4.2 H-Equal

Logical if-and-only-if, on which H-IFF is based is a special case of equality. A simple generalization of the base functions to H-IFF, enables the definition of H-Equal - Hierarchical-Equality.

$$\boldsymbol{t\text{-}equal}(\{s_1, s_2, ...s_k\}) = \begin{cases} s, & \text{if for all i, } s_i = s, \\ \text{null, otherwise.} \end{cases}$$

$$\boldsymbol{f\text{-}equal}\ (s) = \begin{cases} 1, & \text{if } s \neq \text{null}, \\ 0, & \text{otherwise} \end{cases}$$

Firstly, ***t-equal*** is defined for any size alphabet (and null), rather than the binary case of ***t***. Secondly, ***t-equal*** enables us to extend the definition of H-IFF to a larger number of sub-blocks per block. H-IFF is simply the special case of H-Equal where K=2 and S=2. (The size of the alphabet, S, as we use it here - means that there are S symbols in addition to null).

The sections for H-Equal shown in Figure 1c include the section through H-IFF, K=2, as well as K=4 and K=8 for a 64-bit problem. S and K give us two parameters for varying the difficulty of each sub-problem in the overall problem. Of course, we can also vary the overall size of the problem, N (N must be an integer power of K).

The number of solutions to each block in H-Equal is S. And the number of possible combinations of symbols in each block is S^K. We will expect therefore that the difficulty of a problem will increase as S or K increase as in either case the ratio of solutions to possible combinations decreases. All other things being the same, an increase in the difficulty of the base problems will increase the difficulty of the overall problem. However, if we keep N fixed then varying K has a side-effect. Specifically, the number of hierarchical levels in the problem decreases. We will expect that the difficulty of the

[2] Any string with a "01" or "10" block in H-IFF is not a local optima as it can be mutated to "00" or "11", which confers higher fitness, in one bit-flip. However, any string consisting of concatenations of equal pairs ("00"s and "11"s in any order) are local optima since all one-bit mutations decrease fitness. There are 32 bit-pairs in a 64 bit problem, and two kinds of pair, therefore there are 2^{32} local optima.

overall problem will increase with the number of hierarchical levels. Thus the effect of increasing K, although it surely increases the difficulty of each sub-problem, decreases the levels of recursion, assuming N is constant. S, on the other hand, provides a more clear-cut increase in difficulty, as our experimental results will show.

4.3 Bias

H-IFF uses competing schemata, like those found in deceptive functions, in a hierarchically consistent fashion. Following the lead provided by deceptive functions we may bias one of the two competing schema, that is, define one to have a lower fitness contribution than the other. Deb and Goldberg [1992] analyze deception in a trap function in terms of the ratio of fitness contributions for the desired schema and the competing schema. Their analysis concerns only a single trap function rather than a hierarchical structure, and it is assumed that only one of the two competing schemata is the *real* solution, the other is merely a distraction. However, we can still investigate the effect of this ratio in H-IFF. ***f-bias,*** given below, operates on a binary alphabet and K=2, as per the original H-IFF, but unlike the canonical H-IFF the two competing solutions for each block are not rewarded equally. Without loss of generality, we will keep the fitness of the ones blocks at 1, and decrease the fitness of the zeros blocks to some value B, [0, 1]. This will have the effect of making one of the previous two global optima into a unique optimum and depressing the other. We shall refer to this depressed point as the *global complement* since it is the bit-wise complement of the global optimum. Hierarchical consistency will dictate that the ratio of the fitness of the global complement to the fitness of the global optimum will be the same as the ratio of the competing and preferred solutions to each block - i.e. B (this is seen in the landscape sections that follow). Thus we will also refer to the bias value, B, as the *competition ratio*.

$$\textbf{\textit{f-bias}}(a)=\begin{cases} 1, & \text{if } a=1,\\ B, & \text{if } a=0,\\ 0, & \text{otherwise.}\end{cases}$$

Since the preference for schemata that contain the global optima is expressed at all levels we expect that biasing will make the problem easier. Figure 2a shows sections through the H-IFF landscape for various values of B. We see that as the competition ratio approaches 0, where there are no competing schemata, the problem shows no local optima. This case is similar to the Royal Roads problem mentioned earlier, R2, where only one variety of block is rewarded but there are bonuses for pairs and additional bonuses for fours, etc. Intermediate values of B indicate that, because ones are favored consistently throughout all levels, the fitness landscape is merely tilted toward all ones.

4.4 H-XOR and biased H-XOR

When exploring the space of hierarchically consistent problems, it is important to ensure that the transform of a block is a non-separable function of its arguments. If this were relaxed then, since the functions are hierarchically consistent, the entire problem would be separable. Logical if-and-only-if, the base function of H-IFF, is one of only two non-separable functions of two bits that returns a value in the same alphabet. The other is logical exclusive-or, XOR. Other functions have either only one solution or are dependent on only one variable. XOR is the exact negation of IFF, and as such, substituting a transform function based on XOR instead of IFF, to produce H-XOR, yields nothing interesting, in itself. H-XOR, resulting from ***t-XOR*** defined below, has exactly the same properties as H-IFF except the solutions are recursively dissimilar at all scales instead of similar at all scales. For example, solutions for an 8-bit H-XOR are “10010110” and “01101001”.

$$\textbf{\textit{t-XOR}}(a,b)=\begin{cases} 1, & \text{if } a=1 \text{ and } b=0,\\ 0, & \text{if } a=0 \text{ and } b=1,\\ \text{null}, & \text{otherwise.}\end{cases}$$

However, ***t-XOR*** is more interesting when combined with ***f-bias***. The combination produces a problem where 1s are favored over 0s but, since 0s are also required, the GA cannot be permitted to converge on just 1s. The result of varying the competition ratio in H-XOR is indicated in Figure 2c.[3] We see that although B=1 in H-XOR is the same as B=1 in H-IFF the affect of competition ratios less than one is quite different. One interesting feature is that the fitness of the global-complement for B=0 is not zero, as it is in H-IFF, but approximates an average value of the section. Also, there are certainly still local optima.

5 Experimental Results

We shall now begin an experimental exploration of the variants defined above. We have identified the following parameters for affecting the difficulty of hierarchically consistent problems:

- **N**, size of the problem, i.e. number of bits (or symbols).
- **S**, alphabet size.
- **K**, number of sub-blocks per block.
- **B**, bias, or competition ratio (ratio of fitness contributions of competing schema and preferred schema).

We have also defined the variation H-XOR where the effect of bias is not likely to make the problem easy in the same way as it does in H-IFF.

5.1 Algorithms

The basic GA we employ is the same in all cases (and the same as that used in [Watson et al 1998]): A generational GA with a population of 1000; exponentially scaled rank-based selection (scaling factor p=0.01); elitism of 10% (best 10% are transferred to the next generation unchanged); two-point crossover applied with a probability of 0.3; bit-wise mutation with a probability of 2/64 of assigning a new random value to each gene.

---

3 Since the global optima for H-XOR are the recursively dissimilar strings mentioned above, the sections in Figure 2c substitute the beginning of the global-complement “10010110...” into its global optimum “01101001...”.

In our earlier work we demonstrated that a regular GA succeeds easily on H-IFF with the proviso that diversity in the population is maintained. Here, in addition to this basic GA, we also try three augmentations concerned with promoting diversity.

One method used is to share fitness throughout the population according to bit-wise similarity i.e. promoting coverage of both alleles at each locus. This is similar to fitness sharing as defined by Deb & Goldberg [1989] but it is implemented using a resource-depletion model. This model was introduced by Watson et al [1998] but in that work, as in the second variant utilized here, a resource is associated with each solution to every block at every level in the hierarchy rather than only to each bit. The resource model is a form of implicit fitness sharing - rather than limit available fitness to the best individual (in a sample of the population) for each sub-problem, the resource model gives most fitness to the first individual to solve a sub-problem and less to the second, etc. This is a more suitable form of fitness sharing for problems where the sub-problems do not have 'degrees' of solution. The last variant is a coevolutionary model. Here each individual is rewarded only for those blocks it solves that a co-evaluated individual does not solve.

Both of the latter two methods use domain knowledge in their evaluation and they are not intended as general methods of fitness sharing. However, they are illuminating in examining the operation of the GA and in particular the requirement for diversity in the population. In summary, the algorithms implemented are as follows:

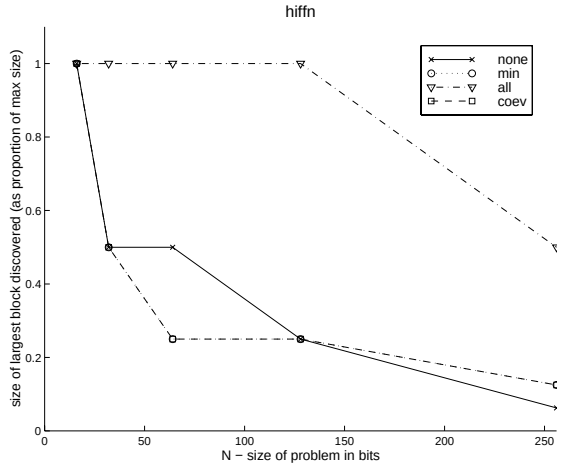
"none" - Basic GA as above with no fitness sharing.

"min" - GA with resource-based fitness sharing only at the bit level.

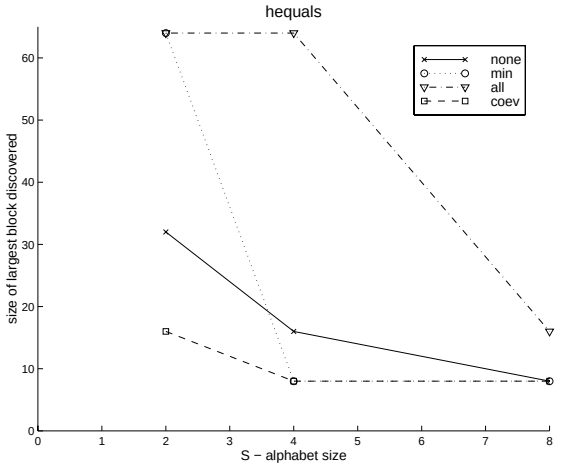
"all" - GA with resource-based fitness sharing for all blocks at all levels.

"coev" - GA with co-evaluation of individuals.

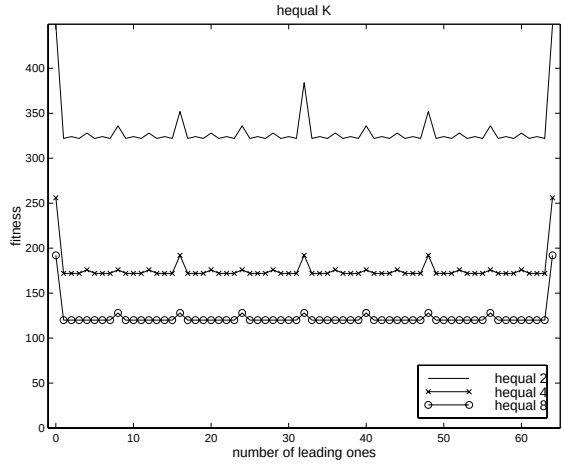
In all cases the algorithms are run for 300 generations and we examine the best string in the last generation. Since the fitness of strings changes in the different variants of the problem it is not meaningful to compare fitnesses. Instead we shall compare the sizes of the biggest all-ones block discovered. In the case where we vary N, we shall show the size of the largest block discovered as a proportion of the largest possible block i.e. size/N. And in the case where we vary the alphabet size S, we will show the size of the biggest block of all the same symbol.



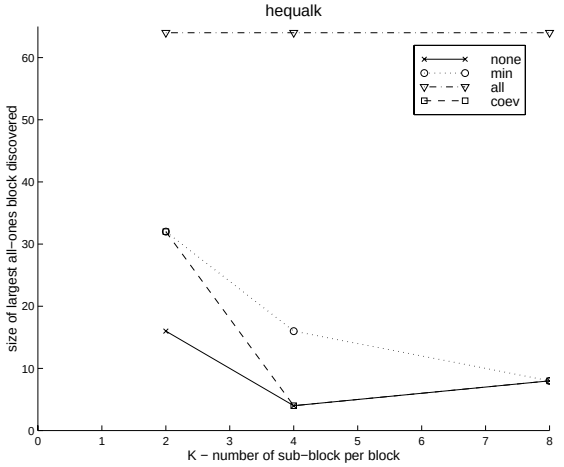
a) performance vs. N, size of problem in bits, (S=2, k=2).



b) performance vs. S, size of alphabet, (N=64, k=2).



c) sections through H-Equal, various K.



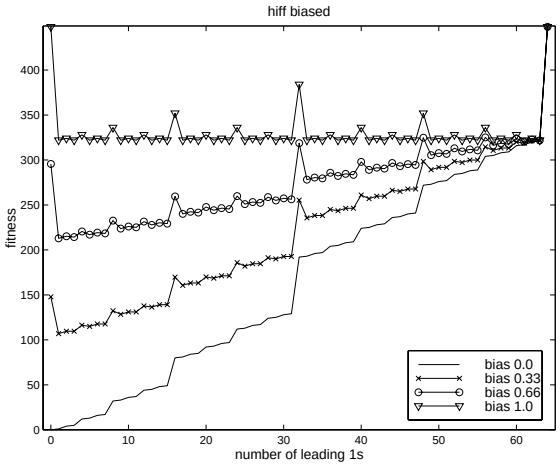
d) performance on H-IFF with various K (S=2, N=64)

Figure 1: Varying N, problem size (in bits), S alphabet size, K number of sub-blocks per block. See text for details.

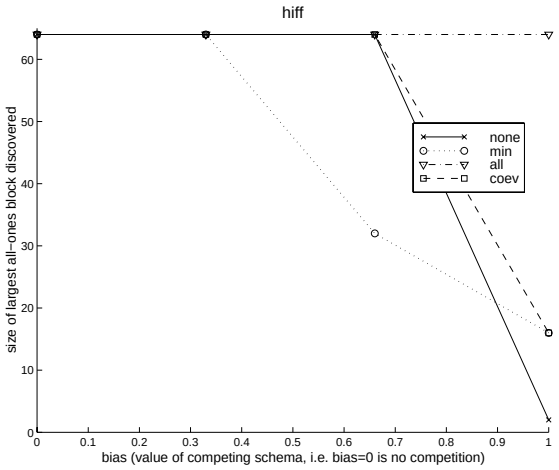
5.2 Results

Figure 1 shows the performance of the four algorithms on various N, S and K, (in addition to the sections through various K, mentioned earlier.) Figure 1a shows performance on H-IFF for N=16, 32, 64, 128 and 256, (K=2, S=2). The first observation is that there is a general decline in performance with increasing N, as expected. Also not surprising is that the algorithm with the strongest fitness sharing performs best. In fact, it succeeds for N up to 128. However, the other algorithms fair poorly; they only succeed reliably for N=16. We shall have to make the problem easier than the canonical H-IFF to see adequate performance in these algorithms that have weaker diversity maintenance. Figure 1b shows performance on H-Equal for S=2, 4 and 8 (N=64, K=2). Again it is not surprising that increasing S decreases performance. Again we see that "all" resources performs best. Figure 1d, performance on H-Equal for various K (S=2, N=64), is more interesting. It shows that when diversity in the population is maintained, increasing K up to 8 does not prevent the GA from finding the global optimum. However, when diversity is not maintained increasing K seems to increase problem difficulty - though the other algorithms performed poorly in any case.

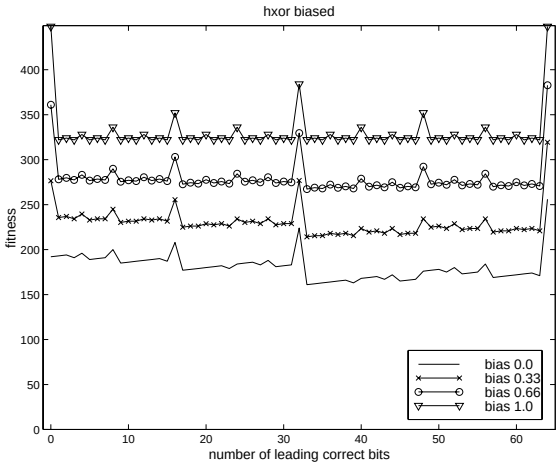
Figure 2 shows the sections and the performance of H-IFF and H-XOR for various competition ratios, B=0.0, 0.33, 0.66 and 1.0. Figure 2b shows the performance on biased H-IFF. Recall that B=1 is the same as the original H-IFF and B=0 means that there are no competing schema. Figure 2b shows that the problem is indeed easier at low values of B; even the GA with no fitness sharing succeeds for low values of the competition ratio. Only as B approaches one and the competing schemata confer equal fitness contributions does the interdependency in H-IFF make the problem hard for the GA, and resolve the abilities of the four GA types. Figure 2d shows the performance on H-XOR with various competition ratios. We see that the effect of changing B in H-XOR is quite different from the effect of changing B in H-IFF. Specifically, even B=0 does not make the problem easier for algorithms that do not maintain diversity - though the results are somewhat erratic with the data collected to date.



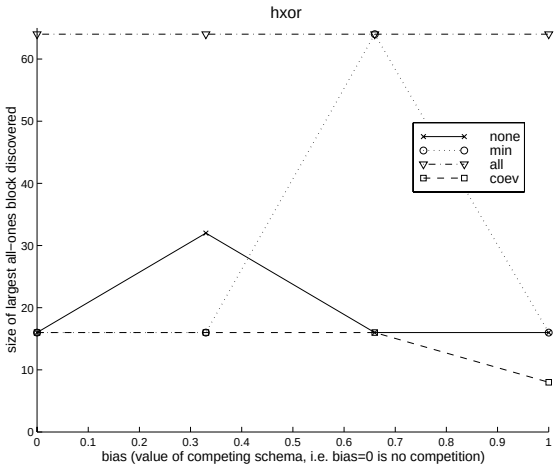
a) sections through H-IFF with various bias



b) performance on H-IFF with various bias



c) sections through H-XOR, various bias



d) performance on H-XOR, various bias

Figure 2. Varying the competitive ratio, or Bias, in H-IFF and H-XOR landscapes. See text for details.

These experiments indicate that the basic GA without diversity-maintenance can succeed on hierarchically consistent problems in the case where simple biasing means that only one type of block is required. However, in the limit where there are no competing schemata, there are also no local optima and therefore this class of problem can also be solved by a hill-climber. As competing schemata become more significant, some form of diversity maintenance mechanism is required to prevent premature convergence.

Hierarchically consistent problems are not delineable into separable functions so we cannot bound their difficulty in the simple way that we can for concatenated trap functions. However, the number of sub-blocks per block, K, provides an alternate measure of problem difficulty. K and S, together with N, assist us in parameterizing hierarchically consistent problems.

We have also seen that varying the ratio of the fitness contributions of the optima affects problem difficulty. But, the effect of this biasing on the whole problem is dependent on the way in which solutions from one level transfer to solutions at the next. In H-IFF, as in the trap functions, when the value of competing schemata is depressed the whole problem becomes easier. However, in H-XOR, we treat both of the competing schemata as required parts of the solution (instead of treating one as a simple distraction as in trap functions). In this case we find that changing the competition ratio does not affect the overall problem in the same way.

It seems likely that a treatment analogous to that provided by Deb and Goldberg [1992b] may yield a critical ratio of bias in H-IFF and H-XOR, and more illumination than these preliminary explorations.

6 Conclusions

Unless we have a good reason to suppose that a problem is different in kind at different scales of resolution, the most parsimonious assumption is that it is the same class at all scales. This paper has defined the concept of hierarchical consistency and we have discussed the shortcomings of existing building-block style test functions in light of this concept. We have introduced several alternative problems which offer various features of difficulty in a hierarchically consistent manner and begun to "re-dimensionalize" problem difficulty for the hierarchically-consistent framework.

Acknowledgments

We are grateful to the members of DEMO at Brandeis, and Alden Wright, for useful discussion and insight.

References

Altenberg, L, 1995 "The Schema Theorem and Price's Theorem", FOGA 3, editors Whitley & Vose, pp 23-49, Morgan Kaufmann, San Francisco.

Deb, K & Goldberg, DE, 1989, "An investigation of Niche and Species Formation in genetic Function Optimization", ICGA3, San Mateo, CA: Morgan Kaufmann.

Deb, K & Goldberg, DE, 1992, "Sufficient conditions for deceptive and easy binary functions", (IlliGAL Report No. 91009), University of Illinois, IL.

Deb, K & Goldberg, DE, 1992b, "Analyzing Deception in Trap Functions", in Whitley, D, ed. FOGA 2, Morgan Kaufmann, San Mateo, CA.

Forrest, S & Mitchell, M, 1993, "Relative Building-block fitness and the Building-block Hypothesis", in Whitley, D, ed. FOGA 2, Morgan Kaufmann, San Mateo, CA.

Forrest, S & Mitchell, M, 1993b, "What makes a problem hard for a Genetic Algorithm? Some anomalous results and their explanation" Machine Learning 13, pp.285-319.

Goldberg, DE, 1989, "Genetic Algorithms in Search, Optimisation and Machine Learning", Reading Massachusetts, Addison-Wesley.

Goldberg, DE, Korb, B, & Deb K, 1989 "Messy Genetic Algorithms: Motivation, Analysis and First Results" Complex Systems 1989, 3, 493-530.

Holland, JH, 1975, "Adaptation in Natural and Artificial Systems", Ann Arbor, MI: The University of Michigan Press.

Horn, J, and Goldberg, DE, 1994, "Genetic Algorithm Difficulty and the Modality of Fitness Landscapes" in FOGA 3, Morgan Kaufmann, San Mateo, CA.

Jones, T, 1995, Evolutionary Algorithms, Fitness Landscapes and Search, PhD dissertation, 95-05-048, University of New Mexico, Albuquerque. pp. 62-65.

Mitchell, M, Forrest, S, & Holland, JH, 1992, "The royal road for genetic algorithms: Fitness landscapes and GA performance", Procs. of first ECAL, Camb., MA. MIT Press.

Mitchell, M, Holland, JH, & Forrest, S, 1995, "When will a Genetic Algorithm Outperform Hill-climbing?" to appear in Advances in NIPS 6, Morgan Kaufmann, San Mateo, CA.

Watson, RA, Hornby, GS, & Pollack, JB, 1998, "Modelling Building-Block Interdependency", PPSN V, Eds. Eiben, Back, Schoenauer, Schweffel: Springer.

Watson, RA, & Pollack, JB, 1999, "Incremental Commitment in Genetic Algorithms", In Banzhaf, W, Daida, J, Eiben, AE, Garzon, MH, Honavar, V, Jakiela, M, & Smith, RE eds. GECCO-99. San Francisco, CA: Morgan Kaufmann.

Whitley, D, Mathias, K, Rana, S & Dzubera, J, 1995, "Building Better Test Functions", ICGA-6, editor Eshelman, pp239-246, Morgan Kaufmann, San Francisco.