Hierarchically Consistent Test Problems for Genetic Algorithms: Summary and Additional Results

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

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

Abstract

This paper gives additional data for experiments presented in previous work on hierarchically consistent test problems. The experiments utilize Hierarchical-if-and-only-if (H-IFF), the basic example of a hierarchically consistent building-block problem, and several variants of H-IFF designed to enable the difficulty of the function to be 'tuned'. We review the H-IFF function and its variants, and give data showing the performance of the regular GA and a fitness-sharing GA for various parameters affecting difficulty.

1 Introduction

The Building-Block Hypothesis [Holland 1975, Goldberg 1989] suggests that the 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 [Mitchell et al 1992, Forrest & Mitchell 1993], have been somewhat perplexing [Forrest & Mitchell 1993b]. Other building-block test problems, for example the concatenated deceptive functions [Goldberg et al 1989], use only a single level of building-blocks and thus depart from the original recursive aspects of the hypothesis.

Demonstrating the recursive building-block assembly described by the building-block hypothesis requires the use of a recursive building-block problem. The Hierarchical-if-and-only-if (H-IFF) function [Watson et al. 1998] provides a principled hierarchical structure where any successful algorithm must move from searching combinations of bits, to searching combinations of low-order schemata, to searching combinations of higher-order schemata, and so on. This type of problem exemplifies the abilities of the GA as described by the building-block hypothesis.

Our previous work [Watson & Pollack 1999] has shown how to separate the recursive structure of H-IFF from the specifics of the problem so that we may generalize the function and define variations of the problem without loosing the principled structure. We identified various parameters that affect the difficulty of the problem and conducted initial experiments. The data presented in the original work was preliminary and shows some anomalies. This paper presents results from re-running those experiments for 30 runs for each data point and shows the effects with more clarity.

The following section reviews the definition of H-IFF and the parameters by which it may be generalized. Section 3 describes other variations of the problem that affect difficulty. Section 4 describes the experiments and gives the new results.

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

In a hierarchically consistent problem, finding the solution to the whole problem given the solutions to its constituent sub-problems, is the same class of problem as finding the solution to a sub-problem given the solutions to *sub*-sub-problems.

Definition: A problem is *hierarchically consistent* if for some ***K***, the problem of finding good schemata of order ***L***, given good schemata of order ***L***/***K***, is of the same class for all ***L***.[1]

H-IFF is our basic example of a hierarchically consistent building-block problem. 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. Each resultant sub-string constitutes a building-block and confers a fitness contribution equal to its size if all the bits in the block have the same value - either all ones or all zeros. The fitness of the whole string is the sum of these fitness contributions for all blocks at all levels.

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

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. Each block, either ones or zeros, represents a schema that contains one of the two global optima at all-ones or all-zeros. Local optima in H-IFF occur when

[1] [Watson & Pollack 1999] uses "length L" in this definition but "order L" is more correct.

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 - 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". However, although local optima and global optima are distant in Hamming space, they may be close in recombination space [Jones 1995] given that the population contains both "11110000" and "00001111" individuals. Thus H-IFF exemplifies the class of problems for which population-based recombinative algorithms are well-suited.

Our previous work [Watson & Pollack, 1998] showed that H-IFF is easy for a GA to solve given that diversity in the population is maintained and genetic linkage is tight. Diversity maintenance methods are addressed in this paper, and algorithms to address poor linkage are addressed in [Watson & Pollack, 1999b].

2.1 Generalizing H-IFF

Although H-IFF is a simple function in itself, and easy to describe, it is desirable that we be able to generalize the function - to create variants that have different properties and different degrees of difficulty. To do this will require us to separate different aspects of the function.

Notice that in H-IFF there are two solutions to each block, all-ones and all-zeros, and these two varieties of solution receive equal fitness contributions. Only compatible varieties of block form solutions to the blocks at the next level. Thus, although the fitness contributions are indistinguishable, these varieties of block are different in terms of how they interact with one another. We must therefore define the interactions of blocks separately from their fitness contributions. Additionally, we wish to separate the specifics of the problem from the general recursive structure. These two separations produce the four functions given in Figure 1.

f and ***t*** are base functions that define the specifics of the problem in terms of a single block. ***F*** and ***T*** are recursive construction functions that use ***f*** and ***t*** to define a consistent function for a whole hierarchy of blocks. In fact, these functions are a generalization of H-IFF that permits any number of sub-blocks per block, ***K***, and any alphabet. The solution to any block is ***K*** symbols of the same type hence, the overall function may be called H-Equal. H-IFF is simply the special case of H-Equal where blocks are assembled pairwise, and the alphabet is binary.

t defines the useful blocks to be any block where all the symbols are the same, and it returns the variety of block that has been formed. For any other case it returns "null" representing a non-solution. ***f*** then states that any non-null symbol is desirable. ***T*** uses ***t*** to recursively determine whether a block of any size is a solution. And finally, ***F*** gives the overall fitness of a string by summing the fitness contributions of all blocks scaled for their size.

	General recursive functions	Problem specific functions
Fitness contributions	$F(B)=\begin{cases} f(B) & if\,\lvert B\rvert=1, \\ \lvert B\rvert\, f(T(B))+\sum_{i=1}^{k} F(B^i) & otherwise. \end{cases}$	$f(a)=\begin{cases} 1 & if\, a \in \mathrm{A}, \\ 0 & otherwise. \end{cases}$
Building-Block interaction	$T(B)=\begin{cases} b_1 & if\,\lvert B\rvert=1, \\ t(T(B^1),\ldots,T(B^k)) & otherwise. \end{cases}$	$t(a_1,a_2,\ldots a_K)=\begin{cases} \alpha & if\,\forall i\,\{a_i=\alpha\}, \\ null & otherwise. \end{cases}$

Figure 1: Separating the general recursive structure of H-IFF from problem specific details, and the interaction of blocks from the fitness contributions of blocks. ***f*** is a base function, ***f***:$\alpha \rightarrow \Re$, giving the fitness of a single symbol, ***t*** is a base function, ***t***:$\alpha^k \rightarrow \alpha$, that defines the resultant symbol from a block of k symbols. α is a symbol from the alphabet *A*. ***F*** and ***T*** are recursive functions that use ***f*** and ***t*** to define the fitness contributions and transforms of a whole hierarchy of blocks. |B| is the number of symbols in B, and B^i is the ith sub-block of B i.e.$\{b_{(i-1)d+1},\ldots, b_{id}\}$ (d=|B|/k).

3 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 and H-Equal 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.

Candidates for varying difficulty in hierarchically consistent problems include: varying the difficulty of the base function by changing the alphabet size, S, or the number of symbols per block, K; changing, ***f***, the fitness contributions of schemata; and changing, ***t***, 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.

3.1 Cross-Sections

To help us in obtaining an intuitive feel for the variations that follow, we 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 2c - this curve 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]

3.2 Varying N, K and S

The sections for H-Equal shown in Figure 2c include the section through H-IFF, K=2, as well as K=4 and K=8 for a 64-bit problem. K, and the alphabet size, S, 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 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 show.

[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.

3.3 Bias

H-IFF uses competing schemata for each building-block, similar to those found in deceptive trap functions [Goldberg et al. 1989]. Following the lead provided by trap functions we may bias one of the two competing schemata, 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 3a 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, where only one variety of block is rewarded - like the second variety of Royal Road problems, R2, 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.

3.4 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".

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

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 3c.[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.

4 Experimental Results

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. [Watson & Pollack 1999] began an experimental exploration of the variants defined above. Some of the results given in that work were inconclusive since the data was collected for only one run of each parameter setting. Here we present results for the same experiments averaged over 30 runs per data point. The original data showed the performance of three varieties of fitness sharing in addition to the basic GA. With the new data it became clear that two of the fitness sharing methods showed no significant improvement in performance in comparison to the GA without fitness sharing. Therefore the methods referred to as "minimum resources" and "co-evaluation" in the original experiments are omitted from the following results, and only the GA with no fitness-sharing and full fitness-sharing are shown.

The basic GA (as in [Watson et al 1998] and in [Watson et al 1999]) is 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.

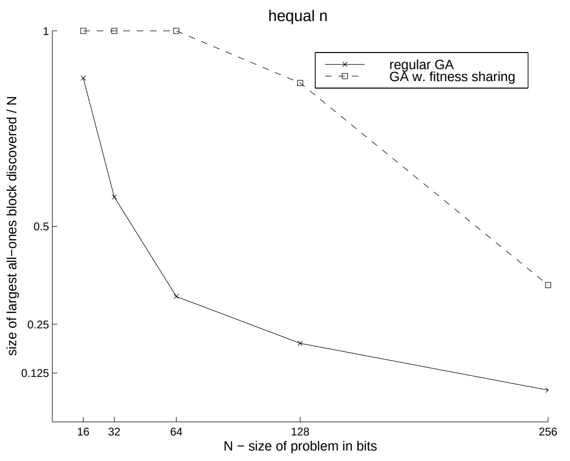
3 Since the global optima for H-XOR are the recursively dissimilar strings mentioned above, the sections in Figure 3c substitute the beginning of the global-complement "10010110..." into its global optimum "01101001...".

Our earlier work demonstrated that a regular GA succeeds easily on H-IFF with the proviso that diversity in the population is maintained. We use a diversity maintenance method using a resource-depletion model [Watson et al 1998] - a resource is associated with each solution to every block at every level in the hierarchy. 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 resources method uses considerable domain knowledge and accordingly it is not intended as general method of fitness sharing. However, it is illuminating in examining the operation of the GA and in particular the requirement for diversity in the population.

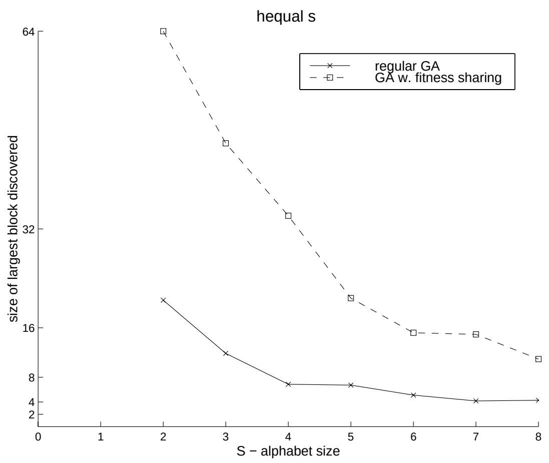
In all cases the algorithms are run for 300 generations and we examine the best string in the last generation. Each point is averaged over 30 runs of the experiment. Since the fitness of strings changes in the different variants of the problem it is not meaningful to compare fitnesses. Instead we shall 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.

4.1 Results

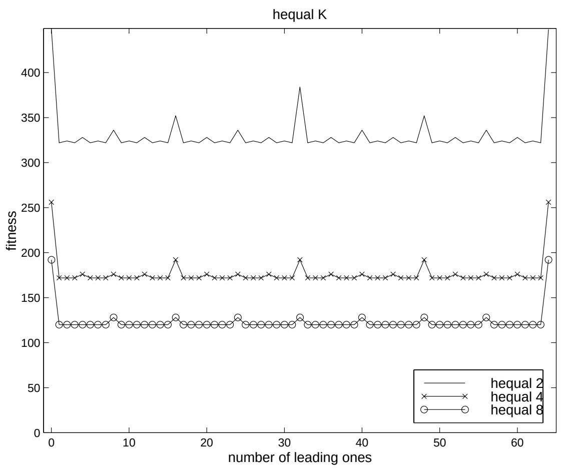
Figure 2 shows the performance of the GA, with and without fitness sharing, on various N, S and K, (in addition to the sections through various K, mentioned earlier.) Figure 2a shows performance on H-IFF for N=16, 32, 64, 128 and 256, (K=2, S=2). The first observation is that there is a general decline in performance with increasing N, as expected. Also not surprising is that the algorithm with fitness sharing performs best. The regular GA performs poorly; it only succeeds reliably for N=16. We shall have to make the problem easier than the canonical H-IFF to see adequate performance in an algorithm without diversity maintenance. Figure 2b shows performance on H-Equal for S=2 through 8 (N=64, K=2). Again it is not surprising that increasing S decreases performance. Again we see that fitness sharing improves performance. Figure 2d, 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 - more than half the runs find the 64 bit blocks with K=8. However, when diversity is not maintained increasing K increases problem difficulty.



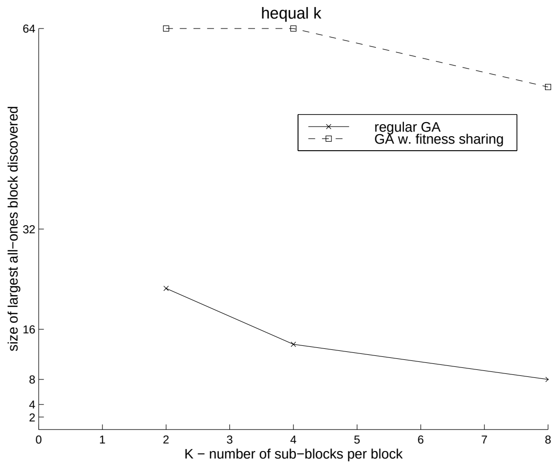
a) performance vs. N, size of problem in bits, (S=2, k=2).



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)

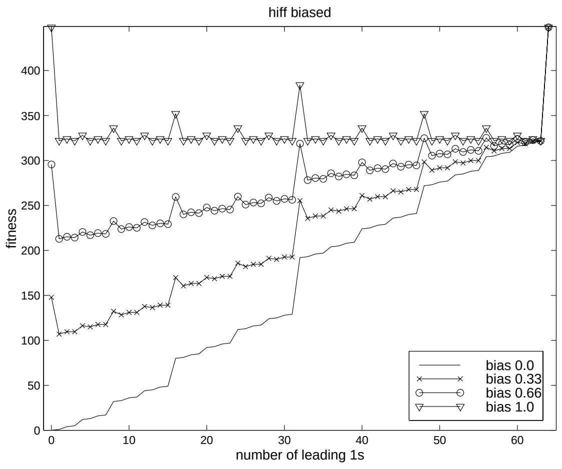
Figure 2: Varying N, problem size, S alphabet size, K number of sub-blocks per block. See text for details

Figure 3 shows the sections and the performance of H-IFF and H-XOR for various competition ratios, B=0.0, 0.2, 0.4, ...1.0. Figure 3b 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 3b 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 distinguish the abilities of GAs with and without fitness-sharing. Figure 3d 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: the GA without fitness sharing is uniformly defeated whereas the GA with fitness sharing is uniformly successful as before.

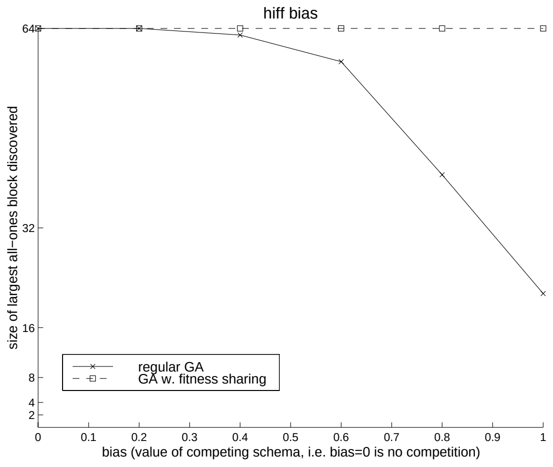
In conclusion, we have introduced several variations of hierarchically consistent problems. These enable us to begin to “tune” the difficulty of hierarchically consistent building-block problems.

The experiments indicate that the basic GA without diversity-maintenance can succeed on hierarchically consistent problems only 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.

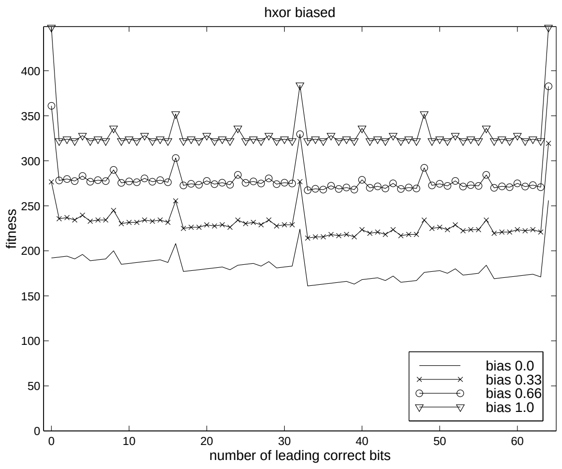
We have also seen that the affect of this biasing on the whole problem is dependent on the way in which solutions from one level transfer to solutions at the next. In H-XOR, we treat both of the competing schemata as required parts of the solution (instead of treating one as a simple distraction as in deceptive functions). In this case we find that changing the competition ratio does not affect the overall problem in the same way.



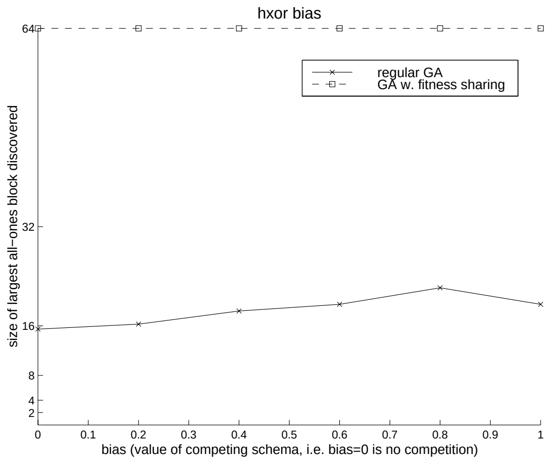
a) sections through H-IFF with various bias



b) performance on H-IFF with various bias



c) sections through H-XOR, various bias



d) performance on H-XOR, various bias

Figure 3. Varying the competitive ratio, or Bias, in H-IFF and H-XOR landscapes. See text for details.

References

Deb, K & Goldberg, DE, 1992, “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.

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.

Watson, RA, Hornby, GS, & Pollack, JB, 1998, “Modeling Building-Block Interdependency”, *PPSN V*, Eds. Eiben, Back, Schoenauer, Schweffel: Springer.

Watson, RA, & Pollack, JB, 1999, “Hierarchically Consistent Test Problems for Genetic Algorithms”, In Angeline, Michalewicz, Schoenauer, Yao and Zalzala, eds., *CEC-99*.

Watson, RA, & Pollack, JB, 1999b, “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.