UC Merced
Proceedings of the Annual Meeting of the Cognitive Science Society

Title
How Learning can guide evolution in hierarchical modular tasks

Permalink
https://escholarship.org/uc/item/8291n256

Journal
Proceedings of the Annual Meeting of the Cognitive Science Society, 23(23)

ISSN
1069-7977

Authors
Wiles, Janet
Tonkes, Bradley
Watson, James R.

Publication Date
2001

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

How learning can guide evolution in hierarchical modular tasks

Janet Wiles (janetw@csee.uq.edu.au)
School of Psychology and School of Computer Science and Electrical Engineering
University of Queensland, Qld 4072 Australia
Bradley Tonkes (btonkes@csee.uq.edu.au)
James R. Watson (jwatson@csee.uq.edu.au)
School of Computer Science and Electrical Engineering
University of Queensland, Qld 4072 Australia

Abstract

This paper addresses the problem of how and when learning is an aid to evolutionary search in hierarchical modular tasks. It brings together two major areas of research in evolutionary computation, the performance of evolutionary algorithms on hierarchical modular tasks, and the role of learning in evolutionary search, known as the Baldwin effect. A new task called the jester's cap is proposed, formed by adding learning to Watson, Hornby and Pollack's Hierarchical-If-and-only-If, function, using the simple guessing framework of Hinton and Nowlan's Baldwin effect simulations. Whereas Hinton and Nowlan used a task with a single fitness peak, ideally suited to learning, the jester's cap is a hierarchical task that has two major fitness peaks and multiple sub-peaks. We conducted a series of simulations to explore the effect of different amounts of learning on the jester's cap. The simulations demonstrate that learning aids evolution only in search spaces in which the simplest level of modules are difficult to find. The learning mechanism explores local regions of the search space, while crossover explores neighborhoods in higher-order modular spaces.

Introduction

This paper addresses the problem of how and when learning is an aid to evolutionary search in hierarchical modular tasks. It brings together two major areas of research in evolutionary computation (EC), the performance of evolutionary algorithms (EAs) on hierarchical modular tasks, and computational models of the role of learning in evolutionary search, known as the Baldwin effect.

We begin with a brief review of modular tasks that have been proposed to explore the performance of evolutionary algorithms, and then briefly describe the Baldwin effect. We then describe a specific task, the *jester's cap*, that incorporates learning into a hierarchical modular task. In many simulation tasks, learning is costly and does not improve the performance of an evolutionary algorithm (French and Messinger, 1994; Mayley, 1996; Pereira et al., 2000). This study is as much an investigation of things that don't learn, as of ones that do.

There are many types of EAs, and the field of evolutionary computation is still determining features of problems that are easy or hard for a particular class of EA, and the conditions under which such algorithms will perform better than other search techniques. In evolutionary computation, characterization of an EA's performance concerns not just optimization per se, but the behaviors of populations as a whole, reflecting their original motivations as models (albeit abstract ones) of real evolutionary processes.

Some of the oldest and most popular techniques for evolutionary search are genetic algorithms (GAs), which use crossover as their major search technique. Originally developed by Holland (1992), their efficacy is thought to be based on groups of genes acting together as modules (or *building blocks*, to use Holland's original terminology), and have been studied extensively since (for general introductions see Goldberg, 1989 and Mitchell, 1996).

A variety of modular tasks have been proposed to study the conditions under which GAs outperform comparable search techniques. The most widely known of these are the Royal Road (RR) problems introduced by Mitchell et al. (1992). However, some forms of hill-climbers were found to easily outperform the GA, and a variety of tasks that incorporate deceptive elements have been defined (s.a., RR4 by Mitchell et al., 1994; HDF by Pelikan and Goldberg, 2000; hdf by Holland, 2000).

An alternative approach to incorporating deceptive elements is to define a fitness function with two or more conflicting maxima. Watson et al. (1998) defined Hierarchical-If-and-only-If (*H-IFF*) as such a function. H-IFF is a simple 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 ends 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 minima (see below for the complete definition).

Before proceeding further with the computational aspects, it is worthwhile considering the relevance of module building to many areas of cognitive science. The role of modules in evolution has long been recognized (e.g., Dawkins, 1986). In evolutionary psychology there is a particularly strong interest in modules, in part due to Tooby and Cosmides (1994) claims that humans have behavioral modules analogous to other mental functions. By studying building block problems, we are considering the types of processes that allow species to evolve varieties of modules, and their combination into complex mental organs. For example, echolocation in bats requires both the ability to emit and to receive high fre-

quency sounds. Each of these abilities has utility as a module in its own right, but the combination provides an additional capacity that goes beyond the cumulative benefit of the independent components.

The Baldwin effect: How learning can guide evolution

Under Darwinian inheritance, the things that an animal learns during its life cannot be passed directly to its offspring via the genotype. However, researchers in the late 19th century (Baldwin, 1896; Morgan, 1896), proposed a way in which learned behaviors could be incorporated into a genome over many generations (i.e., become instinctual). The mechanism is purely Darwinian, and relies on gradual changes in the distribution of genes in a population, as the following rationale explains.

Consider a population of agents comprising a variety of search strategies with initially random starting points and a range of search radii. The starting point and search radius of an agent is its "bias". An evolutionary algorithm that selected for speed in finding a point in the search landscape over many generations would evolve a population of individuals that had starting points close to the fitness maximum and small search radii. That is, behaviors that were initially interpreted as general learning abilities would, over time, become innate. No information about the content of learning is passed from parent to offspring, but in the general variation across the population, some individuals would by chance have starting points closer to the fitness maximum and smaller search radii (i.e., slightly stronger biases). These individuals would have more offspring than those with weaker biases, and in environments with fixed fitness functions, innate behaviors would gradually replace learned ones.

This process is often called the Baldwin effect and has two interesting components. The first is the explanation of how learned behaviors can become innate as described above. The other is the power of learning to augment genetic search to build complex modules. Agents that can search their local environment will be able to explore whole regions of search space in their lifetimes, rather than the single point of their own genotype. In this way, learning can enable an evolutionary algorithm to solve problems that are too costly for genetic search alone.

The first computational simulations of the Baldwin effect were by Hinton and Nowlan (1987). They used a needle-in-a-haystack (NIAH) problem, in which the maximum fitness of an agent corresponded to a genotype comprising all ones. Each gene could be one, zero or question mark. The ones and zeroes were fixed values that did not change during an individual's lifetime. The question marks were learnable genes, which could change during a lifetime. Hinton and Nowlan showed that the zero alleles quickly dropped out of the population and the number of question marks reduced over time. Hinton and Nowlan's study is a landmark in EC because it was the first computational demonstration showing how learning can guide evolution.

Hinton and Nowlan's original simulations have stimulated a considerable body of literature, which is only briefly mentioned here: Belew (1990) replicated and extended the original study, adding changing environments and cultural advantage; Harvey (1993) showed how remnants of residual learning are due to genetic drift; French and Messinger (1994) investigated under what conditions learning supplements genetic search; Mayley (1996) demonstrated how learning is first selected for, then against as evolution progresses; and Mitchell and Belew (1996) discuss issues arising from the original study. A useful reference is the edited volume of papers relating to learning and evolution by Mitchell and Belew (1996), which includes reprints of the original papers by Baldwin (1896), Morgan (1896), and Hinton and Nowlan (1987) as well as many other related studies.

Learning in hierarchical modular spaces

The issues in this study follows from French and Messinger's (1994): Consideration of the circumstances under which is it reasonable to expect that learned behaviors will firstly enhance evolutionary search, and secondly be gradually replaced by genetically specified traits. However, we take an alternative approach and investigate whether learning aids evolution in search spaces that contain competing modules. One way of thinking about this issue is in terms of the difficulty of the search task compared to the operators that are available. In Hinton and Nowlan's NIAH task, the set of twenty ones can be considered as one module, with no intermediate fitness levels for partial results. The search task for the genome is too large to find by populations of size substantially smaller than that of the search space. In this case, learning performs the function of a local search through points close together in Hamming distance. The local search supplements the genetic search, effectively smoothing the search landscape (see top, Figure 1).

The NIAH task is a particularly pathological as it contains no partial information to guide a search process. The majority of tasks of interest in EC and cognitive science have some internal structure, or distinct modules. The most tractable problems that have modular structure are those in which the genes that comprise modules can be selected independently of the settings or global structure of other genes.

As described above, a variety of such tasks have been proposed to explore the functionality of GAs, including the Royal Road problems (Forrest and Mitchell, 1993; Mitchell et al., 1994). The Royal Road problems had only one fitness peak, and hence hill climbing strategies worked well (see Mitchell, 1996 for a summary of the reasons).

The H-IFF problem has the interesting property of symmetry around diametrically opposed fitness peaks with many sub-optimal peaks and consequently many local minima (see bottom, Figure 1).

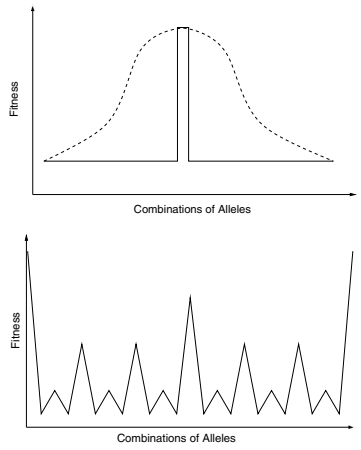

Figure 1: Fitness landscapes in different search tasks. (top) The needle-in-a-haystack task has a single fitness peak, and learning smoothes the search landscape around the peak (adapted from Hinton and Nowlan, 1987, Figure 1). (bottom) A slice through the fitness landscape of H-IFF, showing the multiple fitness peaks and the two maxima at all ones and all zeros (adapted from Watson and Pollack, 1999, Figure 1).

Tasks: H-IFF and the Jester's Cap

H-IFF (Watson and Pollack, 1999) is a function 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 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 optimum bit-strings consist of either all 0s or all 1s so that they are rewarded for building blocks of every size. In the simulations in this paper, we use $2^n = 32$. The evaluation of the H-IFF function is more easily understood by way of example, shown for an 8-bit string in Figure 2.

As described above, the major difference between H-IFF and the more well known Royal Road (RR) 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.

In this paper, we apply the learning-based approach of Hinton and Nowlan's simulations to the H-IFF function. We call this modified version the jester's cap. Specifically, we consider a genome of 32 genes, each of which may be a 0, 1 or ?. During its lifetime, an individual tries to 'learn' the best setting of the ? genes. Each of the ? genes can be set to either 0 or 1, and the resulting bit-string, comprising all 1s and 0s is evaluated with the H-IFF fitness function, described above. We take a very simplistic view of learning (as in Hinton and Nowlan's original simulations), and give the agent N attempts at guessing the best setting. The agent tests N replacements of all of the question marks with random choices of 1s and 0s. After N guesses, the best guess (i.e., that which maximizes fitness) is taken and the fitness of the agent is the H-IFF fitness of that guess. (Unlike Hinton and Nowlan's simulations, there is no scaling of the fitness based on the number of guesses required.) For example, an agent with the genome 0??1 may generate the guesses 0101, 0111 and 0011 which evaluate to 4, 6 and 8 respectively. The highest scoring guess (0011) is taken as the 'learned' setting. However, this 'learned' setting is not passed on during reproduction, it is the initial genome, 0??1 that is used in reproduction.

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

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

Simulation 1: The Jester's Cap

We consider the jester's cap task with three variations of the amount of learning time available to the agents: no learning (replicating the H-IFF task), a small amount of learning ($N = 10$) and a moderate amount of learning ($N = 100$). A population comprising 500 individuals is embedded within a genetic algorithm. In this initial population, 50% of the genes are ?s, 25% are 1s and 25% are 0s, except in the case without learning, where there are no ?s and equal proportions of 1s and 0s. In each generation, the fitness of each of the agents is determined, after learning when appropriate. These fitness values are used to determine the parents for the next generation, those agents with higher fitness being (probabilistically) more likely to be selected as parents than those with lower fit-

nesses. (We used a sigma-scaled roulette algorithm for choosing the parents, see Wiles et al. for further details.) Each pair of parents is used to generate two new offspring using single point crossover (zero mutation). In this recombination technique a 'cut-point' is selected at a random position on the genome. One offspring is generated by joining the genes to the left of the cut-point in parent 1 to the genes to the right of the cut-point in parent 2. The second offspring is formed by the reverse combination.

The idea behind this evolutionary approach is that in the initial population some agents will, by chance, happen to have lower-level modules (or the ability to learn them). These agents will have a slightly higher fitness than the rest of the population, and will be selected as parents more often. When two such agents are paired together for reproduction, it is quite likely that one of the offspring will have two modules, one from each parent. These modules may also combine to form a higher-level module. In later generations, even larger modules can form, so that after a sufficient number of generations the low-level modules that were initially scattered randomly throughout the population have combined in single individuals. The genes of these individuals then begin to dominate the population due to an enhanced fitness, so that every individual has the high-level modules.

These simulations were repeated 100 times for each condition, varying the initial random seed. During the course of a simulation, the mean fitness of the population is monitored. Simulations were run for a maximum of 2000 generations, or until the population converged.

Results

In all three conditions, a sizeable proportion of the 100 populations converged on genotypes of maximum fitness. On average, trials in which agents were allowed either no learning or a moderate amount of learning outperformed those where less learning was allowed, as shown in Figure 3.

Discussion

The genetic operators of crossover are maximally suited to the hierarchical structure of the H-IFF problem. Unsurprisingly, crossover works well on this problem. Learning, which one might expect to perform as well or better, does not match the performance of the genetic operators alone. This result can be explained by considering the way that learning searches the space. Learning in the jester's cap is a mechanism for searching the neighborhood as determined by Hamming distance. This search is only effective for the lowest level of building blocks. At subsequent levels, local and global minima are close in recombination space, but not in Hamming space. At these higher levels, learning merely adds distractions to an otherwise successful algorithm, although adding a sufficient amount of learning can negate any detrimental effects.

In conclusion, learning in this type of hierarchical task is no more effective than genetic search because a way is needed to search module space, not Hamming space. As posed, the jester's cap assumes that low level modules are trivial to find. We next consider a sparse version of the task in which they are not so readily revealed.

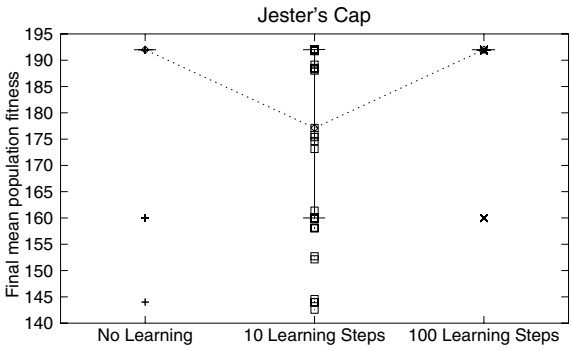

Figure 3: Performance of populations on the jester's cap task under three conditions. Each point represents the final mean fitness of a population. Error bars show first and third quartile and the medians are linked. Note that in the no learning (left) and moderate learning (right) conditions, the error bars are obscured because first, second (median) and third quartiles are all equal. Note also that many populations have identical mean fitness and tend to cluster around a set of discrete values because of the nature of the H-IFF function.

Simulation 2: The Sparse Jester's Cap

In the jester's cap simulations, rewards were given for modules of all levels (i.e, 1, 2, 4, 8, 16 and 32). In the sparse jester's cap, we consider rewarding only a subset of the levels. For example, in Figure 2 the blocks of size 2 may not contribute to the overall fitness of the solution. This modification allows us to vary the nature of the task from the maximally hierarchical H-IFF function (where all levels rewarded) to the NIAH function (where only the highest level rewarded). Varying the rewarded levels of the H-IFF function changes both the ease with which the initial modules can be found, as well as the ease with which the lower-level modules may be combined into the next higher-level of module. In the simulations in this section, only the building blocks of size 1, 16 and 32 are rewarded. With these choices of building-block and population sizes the smallest non-trivial modules (those of size 16) are difficult to find (*cf.* Hinton and Nowlan's simulations where the module is of size 20). It is thus expected that learning will substantially assist for the low-level modules. We repeated the first series of simulations using this alternative fitness function.

Results

Not surprisingly, the populations evolved using the sparse jester's cap fitness function fared substantially worse than those evolved with the (standard) jester's cap,

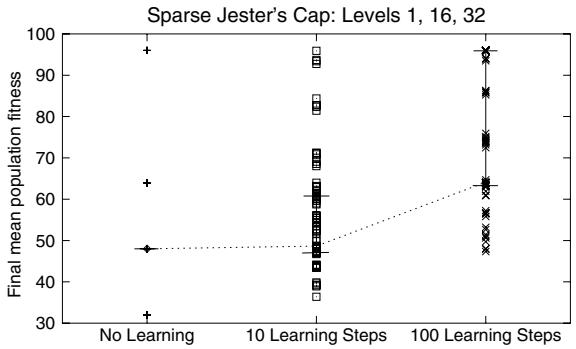

Figure 4: Performance of populations on the sparse jester's cap task under three learning conditions. Each point represents the final mean fitness of one population. Error bars show first and third quartile which are again obscured in some cases. Most populations with no learning (left) converged on a final fitness of 48, corresponding to one module. Populations in the moderate learning case converged on genomes giving a variety of fitness values, indicating some amount of residual learning in the genome (right). With a small amount of learning (center), performance was marginally improved over the no-learning case, although again residual learning remained.

as shown in Figure 4 (note that the y-axes differ between Figure 3 and Figure 4). In the condition of no learning, most populations (83 of 100) found a single module. With a small amount of learning, populations converged on marginally better solutions on average. In this condition, many residual question marks remained in the final populations. As a result, the individuals from these populations could only find modules with some degree of chance, causing the observed scattering of results in Figure 4. With moderate learning, some populations converged on the optimal genomes, others converged on genomes that gave agents the potential of finding the optimal solutions (i.e., there was again residual learning), while others converged on poor genomes. However, the populations with a moderate degree of learning, on average, outperformed the populations in the other learning conditions.

Discussion and Conclusions

This paper addressed the problem of how and when learning is an aid to evolutionary search in the jester's cap, a hierarchical modular task. Simulation 1 showed that evolutionary search could efficiently find the optimal solution with no learning. The addition of a small amount of learning detracted from this performance. A moderate amount of learning had no benefit, but did not detract either. It turns out that H-IFF is not a difficult task for a GA at the levels of complexity studied in this paper.

The main issues in reaching the highest levels of performance on H-IFF relate to maintaining a diversity of modules at intermediate levels. The population size of 500 in these simulations was clearly adequate for maintaining this diversity. Adding learnable alleles increases the search space, without a reciprocal benefit in assisting search.

Simulation 2 showed that the sparse jester's cap (1,16,32), a problem intermediate between H-IFF and NIAH, was not amenable to evolutionary search by the GA. The majority of populations only found a single module (fitness level 48). The smallest module involved 16 bits, and the likelihood of finding two such modules in any one population before convergence was minimal.

In contrast to Simulation 1, in Simulation 2 a small amount of learning (ten steps) marginally improved the success with which populations discovered modules. Ten learning steps is sufficient to effectively search four to five learnable genes, and in this case, learning clearly did provide a reciprocal benefit that more than compensated for the increase in the search space.

Increasing the learning from ten to 100 steps substantially improved the success of populations. All populations found at least one module, 75% found two modules, and at least 25% reached optimal performance. One hundred learning steps is sufficient to search six to seven learnable genes. Although this is only two more than searched by ten learning steps, it had a demonstrable effect on performance.

Simulation 2 demonstrated that in the sparse version of the jester's cap, learning is required to discover the modules, as in NIAH. As in Hinton and Nowlan's simulations, populations are unlikely to find high-level modules by crossover alone. Learning is able to guide the population towards finding the low level modules, and then crossover combines them. However, the performance is still not optimal, and room for improvement remains.

In conclusion, there appears to be a role for learning in situations where crossover is an ineffective search technique. Crossover searches module space whereas learning searches Hamming space. In tasks such as the jester's cap there is very little need for searching Hamming space and the majority of optimization can be effectively performed in module space. In this task, Hamming search is useful only at the lowest-level of module. For higher-level modules crossover searches through combinations of peaks, rather than traversing the troughs between them (Figure 1). Local search provides the wrong operator for preserving and improving fitness because it spends too much time in the troughs of fitness space. In the sparse jester's cap, modules must be discovered before searching module space becomes a viable approach. The difficulty in finding these modules necessitates local search.

NIAH and H-IFF may be viewed as being on alternative ends of a spectrum. In the former, the (single) module is difficult to find as there is no partial feedback to guide search. Learning is required to act as a proxy for this partial feedback. In the latter, modules abound so the important factor is not finding the modules, but discovering how to put them together. Consequently, the

best search process is one that searches through combinations of modules rather than searching for the modules themselves. In this case, learning is merely a hindrance. The sparse jester's cap represents an intermediate point on the spectrum. Modules are sufficiently difficult to find so that learning is required to give partial feedback in the search for the lowest-level modules. Once the modules have been found, recombination can function.

Acknowledgements

We thank Rik Belew and two anonymous reviewers for their constructive feedback on this paper. This research has been supported by a CSEE scholarship to JRW.

References

Baldwin, J. M. (1896). A new factor in evolution. *American Naturalist*, 30:441–451. Reproduced in Belew, R. K. & Mitchell, M. (Eds.), *Adaptive Individuals in Evolving Populations*. Addison-Wesley, Reading, MA.

Belew, R. K. (1990). Evolution, learning and culture: computational metaphors for adaptive search. *Complex Systems*, 4(1):11–49.

Dawkins, R. (1986). *The Blind Watchmaker*. Penguin Books.

Forrest, S. and Mitchell, M. (1993). Relative building-block fitness and the building-block hypothesis. In Whitley, D., editor, *Foundations of Genetic Algorithms*, volume 2, pages 109–126. San Mateo, CA: Morgan Kaufmann.

French, R. and Messinger, A. (1994). Genes, phenes and the baldwin effect: Learning and evolution in a simulated population. In Brooks, R. A. and Maes, P., editors, *Artificial Life IV*, pages 277–282.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA.

Harvey, I. (1993). The puzzle of the persistent question marks: A case study of genetic drift. In Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 15–22, San Mateo, CA. Morgan Kaufmann.

Hinton, G. and Nowlan, S. (1987). How learning can guide evolution. *Complex Systems*, 1:495–502.

Holland, J. H. (1992). *Adaption in Natural and Artificial Systems*. MIT Press, 2nd edition.

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

Mayley, G. (1996). The evolutionary cost of learning. In Maes, P., Mataric, M. J., Meyer, J.-A., Pollack, J., and Wilson, S. W., editors, *Proceedings of the Fourth International Conference on Simulation of Adaptive behavior: From Animals to Animats 4*. MIT Press/Bradford Book.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA.

Mitchell, M. and Belew, R. K. (1996). Preface to 'How learning guides evolution' by G. E. Hinton & S. J. Nowlan. In *Adaptive Individuals in Evolving Populations: Models and Algorithms*, volume XXVI of *Santa Fe Institute Studies in the Science of Complexity*, pages 443–446. Addison-Wesley.

Mitchell, M., Forrest, S., and Holland, J. H. (1992). The royal road for genetic algorithms: Fitness landscapes and GA performance. *Proceedings of the First European Conference on Artificial Life*, pages 245–254.

Mitchell, M., Holland, J. H., and Forrest, S. (1994). When will a genetic algorithm outperform hill climbing. In Cowan, J. D., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems*, volume 6, pages 51–58, San Mateo, CA. Morgan Kaufmann Publishers, Inc.

Morgan, L. C. (1896). On modification and variation. *Science*, 4:733–740.

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

Pereira, F. B., Machado, P., Costa, E., Cardoso, A., Ochoa-Rodriguez, A., Santana, R., and Soto, M. (2000). Too busy to learn. In *Proc. of the 2000 Congress on Evolutionary Computation*, pages 720–727, Piscataway, NJ. IEEE Service Center.

Tooby, J. and Cosmides, L. (1994). Origins of domain specificity: The evolution of functional organisation. In Hirschfeld, L. and Gelman, S., editors, *Mapping the Mind*, pages 85–116. Cambridge University Press.

Watson, R., Hornby, G., and Pollack, J. (1998). Modeling building-block interdependency. *Parallel Problem Solving from Nature, proceedings of the Fifth International Conference*, pages 97–106.

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

Wiles, J., Schulz, R., Bolland, S., Tonkes, B., and Hallinan, J. Selection procedures for module discovery: Exploring evolutionary algorithms for cognitive science. This volume.