Exploiting Problem Structure in Genetic Algorithms

Scott H. Clearwater and Tad Hogg

Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304, U.S.A.
clearwat@parc.xerox.com, hogg@parc.xerox.com

Abstract

Recent empirical and theoretical studies have shown that simple parameters characterizing the structure of many constraint satisfaction problems also predict the cost to solve them, on average. We apply these observations to improve the performance of genetic algorithms. In particular, we use a simple cost measure to evaluate the likely solution difficulty of the different unsolved subproblems appearing in the population. This is used to determine which individuals contribute to subsequent generations and improves upon the traditional direct use of the underlying cost function. As a specific test case, we used the GENESIS genetic algorithm to search for the optimum of a class of random Walsh polynomials. We also discuss extensions to other types of machine learning and problem solving systems.

Introduction

Several recent studies of NP-hard search problems have shown that easily computable characteristics of their structure determine, on average, their *hardness*, i.e., the cost to solve them with a variety of heuristic search methods [Cheeseman et al., 1991, Mitchell et al., 1992, Williams and Hogg, 1992a, Williams and Hogg, 1992b]. While these results provide insight into the nature of NP-hard problems, there remains the issue of whether they can also be used to improve search methods. If possible, an improvement based on these results would be a significant domain-independent heuristic. At first sight this might appear difficult since the relation between problem structure and hardness only holds on average: the large observed variances indicate that any individual problem instance can deviate significantly from the average behavior. Thus these results cannot be expected to give detailed guidance for sophisticated domain-specific heuristics.

However, some search methods, such as genetic algorithms [Holland, 1975, Goldberg, 1989, Forrest, 1993] (GAs), rely on a statistical sample of search states. Evaluating these states with respect to the overall goal is then used to guide the selection of further states. To the extent that the selection method is able to focus the search toward solution states, on average, these methods can be effective. Such methods are natural candidates for exploiting an improved understanding of average problem hardness.

The novel contribution of this paper is using a theory relating problem structure to hardness to help select individuals within the context of a genetic algorithm, and comparing this improvement with the traditional approach. We do this for a particular class of constraint satisfaction problems. We also discuss how this specific example may be generalized to learning programs and other search problems. Our results suggest that the relation between problem structure and hardness can indeed be exploited to give improved domain-independent heuristics.

Genetic Algorithms for Constraint Satisfaction

Genetic algorithms are a general optimizing search method. They use analogs of evolutionary operators on a population of states in a search space to find those states that minimize the value of a given cost function. Equivalently, they can be viewed as maximizing a fitness function. We used the search space consisting of bit-strings of length μ, commonly employed with GAs. Each particular search problem was defined by a cost function on these states, with a known optimal state. By contrast, the studies of problem structure and hardness have focused mainly on constraint satisfaction problems (CSPs). In these problems, one is given a set of constraints and attempts to find a state in a search space that satisfies all of them (i.e., a solution to the CSP), or prove no such state exists. In our work, we used a simple class of optimization problems that can also be viewed as CSPs, thus allowing for the most direct use of the hardness theory.

Specifically, our cost function can be expressed as a type of Walsh polynomial, i.e., a sum of discrete Walsh functions with coefficients. The Walsh functions form a basis set for functions defined on bit-strings. These polynomials, which are thus much like Fourier series, have been studied previously with GAs [Forrest and Mitchell, 1993]. Each such Walsh function is specified by a bit-string β and maps bit-strings b in the search space to ± 1 as:

$$\psi_\beta(b) = \begin{cases} 1 & \text{if } b \wedge \beta \text{ has even parity (even no. of 1's)} \\ -1 & \text{otherwise} \end{cases} \quad (1)$$

where $\wedge$ is the bitwise AND operator. The number of *1*'s in β is referred to as the *order* of the Walsh function. Example values for an order-2 function are $\psi_{1100}(1110) = 1$ and $\psi_{1100}(1010) = -1$. A function defined on bit-strings of length μ is expressed as a Walsh polynomial as:

$$F(b) = \sum_{\beta=0}^{2^\mu - 1} \omega_\beta \psi_\beta(b) \qquad (2)$$

where ω_β are real coefficients. Here the sum is over all possible bit-strings β of length μ. The *number of terms* in $F(b)$ is defined to be the number of nonzero coefficients ω_β appearing in the sum.

For our experiments, we defined a class of optimization problems by selecting the Walsh polynomials randomly according to specified parameters to correspond to the simplest CSPs studied with the theory. First, we only included Walsh functions of a specified order k. And among these, we selected randomly exactly n terms to include in the Walsh polynomial, i.e., from among the $\binom{\mu}{k}$ Walsh functions of order k, we randomly selected n to use. Second, for each problem we selected a random bit-string B and chose the sign of each coefficient so that $\omega_\beta \psi_\beta(B) > 0$, with the magnitude of the coefficient a random integer in the range $[1, 5]$. With this choice of signs, the maximum value of F is:

$$F_{max} \equiv \max_b (F(b)) = \sum |\omega_\beta| \qquad (3)$$

and is achieved by the state B. Finally, the optimization problem presented to the GA in our experiments was to find a bit-string b which minimized the cost function

$$c(b) = F_{max} - F(b). \qquad (4)$$

The minimum of this cost is zero, so we can readily determine how close to optimal the GA gets.

The optimization problem defined this way can also be viewed as a constraint satisfaction problem (CSP) with a prespecified solution. Specifically, we can view each position in the bit-string as a variable which can be assigned one of two values (0 or 1). Moreover, each term $\omega_\beta \psi_\beta$ in the Walsh polynomial corresponds to a constraint. A given bit-string b satisfies the constraint if and only if $\omega_\beta \psi_\beta(b) > 0$. By our choice of signs for the ω_β the minimum value of the cost function is achieved only when all terms are positive, so a minimum cost state corresponds to a solution to the CSP, i.e., a state in which all constraints are satisfied. In particular, the state B is a solution.

Theory

By viewing the optimization problem as a CSP, we can apply recently developed theories [Williams and Hogg, 1992b] to characterize the difficulty of searching for a solution. In this work, a CSP is characterized by the number of variables (μ), the domain size of each variable (2 for our case of binary variables), and the number and size of the *minimized nogoods* of the constraints. These nogoods are simply those smallest subsets of all possible states in the problem that violate at least one constraint. Their *size* is just the number of variables involved.

μ	number of variables and bits in the state
k	size of constraints = order of Walsh functions
n	number of constraints and terms in Walsh polynomial
m	number of minimized nogoods, $n2^{k-1}$

Table 1. Mapping between theory parameters and the experimental testbed.

For example, consider a Walsh polynomial with one term, $F(b) = 2\psi_{1100}(b)$. The corresponding CSP has μ=4, k=2, and n=1. Since only the non-zero bits are important in determining the value of the Walsh polynomial there are at most two variables we need to be concerned about, in this particular example the first and second positions. For these two variables there are two ways to obtain $\psi_{1100} = +1$ and hence a positive contribution to F since $\omega_{1100} = 2$ is positive (the "goods"), $\{b_1 = 0, b_2 = 0\}$ and $\{b_1 = 1, b_2 = 1\}$. There are also two ways to obtain a -1 (the "nogoods"), $\{b_1 = 1, b_2 = 0\}$ and $\{b_1 = 0, b_2 = 1\}$ where $b_i = s$ denotes the assignment of value s to variable i, i.e., value s appearing at position i in the bit-string b.

More generally, a Walsh function of order k will involve exactly k variables, so the corresponding nogoods will have size k. Moreover, of the 2^k possible assignments to these variables, exactly half will have even parity giving $\psi_\beta(b) > 0$. Thus, for either choice of the sign of the coefficient ω_β, exactly half the assignments will violate the constraint associated with this term, i.e., will be nogood. So each term in the Walsh polynomial will contribute 2^{k-1} minimized nogoods. Moreover, these will be distinct from those contributed by other terms since each term has a distinct subset of the variables in the problem. From this argument we see that our CSP's, in which the Walsh polynomial has n terms, have $n2^{k-1}$ minimized nogoods, all of size k. This mapping between the parameters in the theory and our problem class is summarized in Table 1.

For simple backtrack search, the theory [Williams and Hogg, 1992a, Eq. 10] estimates that the search cost, on average, as a function of μ, k and n is

$$C_{theory} = \frac{\sum_{i=0}^{\mu} c^{g_i}}{\max(1, e^{g_\mu})} \qquad (5)$$

with

$$g_i = i \ln 2 + n 2^{k-1} \ln \left(1 - \left(\frac{i}{2\mu}\right)^k\right). \qquad (6)$$

Note that g_i decreases as more terms are added, i.e., as n increases, and this decrease is more rapid for larger values of i. This means that, for fixed μ and k, C_{theory} first increases with n, eventually reaches a peak at a value $n = n_{crit}$ (which depends on μ and k), and then decreases, as shown in Fig. 1 for parameter values used in some of our experiments.

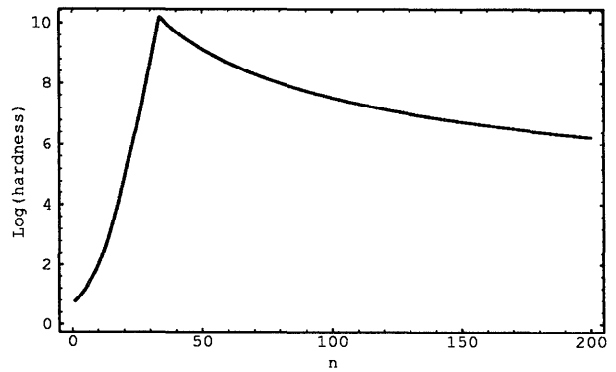

Fig. 1. $\ln(C_{theory})$ versus n, the number of constraints, for $\mu = 25$ and $k = 4$. The maximum hardness occurs for $n = 34$.

A Hardness-Based Cost Measure

As it stands, this theory evaluates the expected search difficulty of an entire problem. Applied in the context of a GA, we need to evaluate the usefulness of different individual states in the population. That is, some states are more likely than others to lead readily to a solution, and hence should have an enhanced number of offspring in the next generation. The traditional fitness function approach simply equates usefulness of a state with the value of the function to be optimized applied to that state, where the function to be optimized may be either maximized or minimized.

We now describe how we applied the theory to evaluate individual states and used this evaluation to define a new cost measure for use with the GA. For an individual state b some constraints in the problem will be satisfied while others are violated. Thus one way to apply the problem hardness measure of Eq. 5 is to view the violated constraints as forming a subproblem and use this subproblem's hardness measure as the hardness associated with the state b. While there are several specific ways one could do this, our particularly simple approach is to replace the value of n in Eq. 5 with the number of constraints (i.e., terms in the Walsh polynomial) that are violated by the state.

The potential advantage of using hardness-based cost is that we can exploit hard and easy parts of the search space and either seek out or avoid those areas as necessary. When applied to GAs this implies we can bias the population in one direction or the other via reproduction operations. For example, if a particular individual had a cost to the right of the maximum hardness peak of Fig. 1 and the next generation had a cost to the left, then we would want to exploit that discovery by heavily increasing that individual's number of offspring for the next generation. Note that this is the case even if the hardness to the left of the maximum is *higher* than the hardness to the right. This is because a problem becomes easier very rapidly to the left of the peak. Thus, in our experiments we sought to decrease the number of violated terms, without considering the weights given to the various terms by the coefficients, unlike the traditional approach.

To do this within the context of a GA, we need to relate the hardness measure we defined for an individual state to an appropriate cost function. In this context we note that the standard cost measure of Eq. 4 monotonically decreases as the state gets close to the optimum. The situation is more complicated for the non-monotonic hardness function of Eq. 5. The region to the left of the maximum peak is monotonically decreasing as n decreases, which is fine. The problem is that the hardness function is monotonically *increasing* to the right of the maximum hardness peak as n decreases. Instead, we need to find a function of hardness which decreases even though hardness itself increases. There are a plethora of ways to define an appropriate cost function. One way that systematically produces better results is as follows. If the state b, with n violated constraints, is "to the right" of the maximum hardness peak, i.e., $n > n_{crit}$, then reducing the cost requires first solving harder problems closer to the peak on the way towards the steeply dropping part of the hardness curve "to the left" of the peak. Thus, to encourage the GA to reduce the number of violated constraints, a function inversely proportional to the hardness can be used. Similarly, when $n < n_{crit}$ a function proportional to the hardness can be used. Moreover, to stress the importance of states to the left of the hardness peak, a rescaling of hardness is done to cause a "stampede" across the hardness peak that might otherwise take a long time. Since the hardness theory is an approximation, the exact location of the peak is uncertain, but on the average it should, like the GA procedure itself, lead to better performance. The cost function we chose was one based on simplicity, namely:

$$c_{hard}(\mu, k, n) = \begin{cases} \ln(C_{theory}) & \text{if } n < n_{crit} \\ 400/\ln(C_{theory}) & \text{otherwise} \end{cases}. \qquad (7)$$

Finally, the cost for given bit-string b was $c_{hard}(b) = c_{hard}(\mu, k, n)$ where n was the number of constraints that conflict with the state b. This cost is minimized when all constraints are satisfied, i.e., when $n = 0$, corresponding

to a solution. To distinguish this use of n for a state b, we use n_0 to denote the total number of terms in the Walsh polynomial of the problem. This remains fixed while the n values associated with a population of states varies as states are modified by the GA.

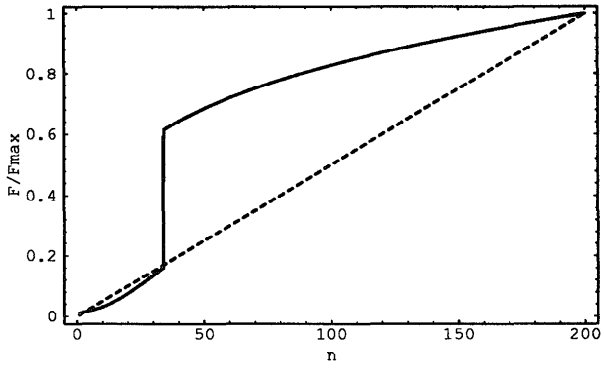

Fig. 2. Comparison of standard (dashed) and hardness-based (solid) cost measures as a function of the number of violated constraints n for $\mu = 25$ and $k = 4$. Both values are normalized to their maximum value in the range considered. Note that the standard measure depends on the values of the coefficients and hence the line represents a hardness averaged over many problems with this structure. Note the abrupt change in the hardness at the transition point $n_{crit} = 34$ which acts to initiate a stampede of individuals across the critical region.

To illustrate our application of the theoretical measure to individual states, we can examine the number of constraints that are violated by individual randomly chosen initial states. It is important to realize that most such states will already satisfy many constraints. Thus, if we assume the constraints are independent, one would expect that the number of terms in conflict with a random initial state would be $n = \frac{1}{2}n_0$. In fact, we observe somewhat more initial conflicts, and further, there is a wide variation in the initial number terms to be solved. This of course leads to a spread in the initial hardness of the problem to be solved.

Experimental Setup

We used a publicly available genetic algorithm called GENESIS for GENEtic Search Implementation System version 5.0 [Grefenstette, 1990]. The program provides a standard genetic algorithm implementation with bit-string or floating point vector representation. The user provides an evaluation function for determining the fitness of the individuals in the population. The user also provides the crossover and mutation rates. There is also an elitist mechanism for guaranteeing that a number of the fittest individuals from one generation will survive to the next. In all the experiments reported here a fitness proportionate reproduction was used where the number of offspring from an individual was proportional to its fitness (which means how well it minimizes cost) in the overall population.

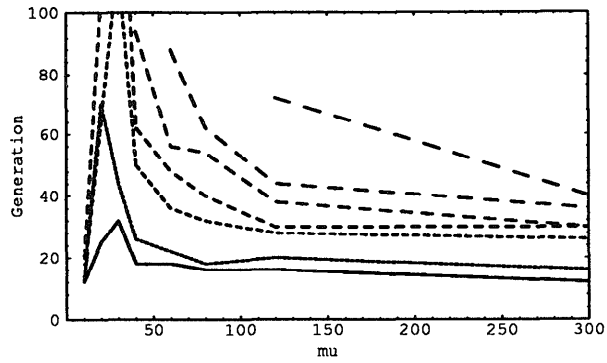

Fig. 3. Generation versus μ with k=4, n=3, using the standard cost measure. The lowest curve (solid black) is the distribution of time to reach 80% of the optimal solution averaged over 10 runs, i.e., the time required to first find a state with $F(b)/F_{max} \geq 0.8$. Similar curves follow upward and become more dashed are for 85%, 90%, 92%, 94%, 96%, and 98%. Curves that do not have certain μ values did not reach the specified level of optimum within 100 generations. This is in qualitative agreement with the theory which states that the hardest problems are in the regions of intermediate μ. A similar easy-hard-easy pattern of problems is seen when n is varied at a fixed value of μ, in qualitative agreement with the behavior shown in Fig. 1.

All of the experiments used 250 individuals with a mutation rate of .02 per bit and a crossover rate of 0.5 per individual. At each generation, the best 10 individuals were guaranteed to survive. Typically 100 runs were made under the same conditions. Each run ran for 100 generations. With our choice of parameters μ, k and n describing the class of problems, the initial hardness was in the overconstrained part of the space. Thus, the individual had to move through the maximum hardness peak to solve the problem. Further, the polynomials were easy enough to be solved in a reasonable amount of time so that decent statistics could be obtained. Still, not every problem was solved within the allotted time. Thus we have a choice of performance metrics: fraction of time the problem was solved within a given number of generations, and fraction of the optimal solution found at a given generation.

Does the Theory Apply to GAs?

Before presenting our evaluation of the new fitness function, we first used the standard GA to see whether the theory applies at all to GAs. Although a relation between problem structure and hardness has been observed for a variety of search methods, it is important to check that it also applies to our problems solved by GAs. More generally, instead of examining the time required to reach an optimal state, we considered near-optimal states as well. Fig. 3 shows how the search cost varies with problem size and closeness to optimality. We see the development of a peak indicating a region of particularly hard problems, in qualitative agreement with the theoretical prediction and empirical observations of other search methods. This generalizes and "explains" more generally

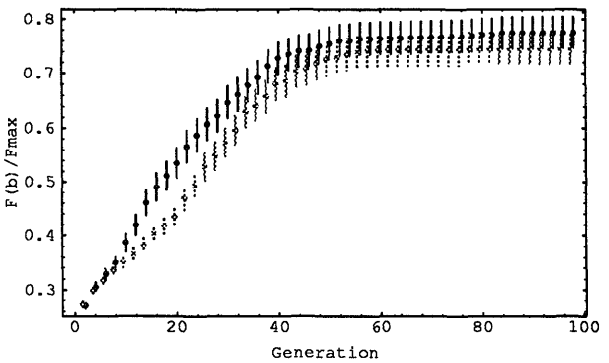

Fig. 4. Fraction of optimum as a function of generation, i.e., the largest value of $F(b)/F_{max}$ for states b in the population. For this plot μ = 25, k = 4, n_0 = 150. Black points are the hardness-based cost results and the gray points are the standard fitness results. The error bars are the standard error of the mean for the generation. All 100 runs are plotted. Much of the displayed variance is due to the different problem instances. A more precise comparison of the two algorithms is given by Hotelling's multivariate paired T test [Johnson and Wichern, 1992]. This test on the differences in the two methods on 400 problems, rejects the hypothesis that they have the same behavior at a significance level of less than 10^{-4}.

two observations [Forrest and Mitchell, 1993] where what naively seems to be a harder problem (since it involves more variables) is actually easier.

Search Performance

We compare the hardness-based cost measure with a standard measure in Fig. 4. Note that in the figure the hardness-based cost led to superior performance. Additionally, the hardness-based cost found the optimum on 63 out of 100 runs versus 58 with the standard fitness. Each run consisted of a randomly generated Walsh polynomial, which was used with both the hardness–based and standard cost functions. By using the same problem instances with both methods we obtain a more discriminating comparison between the methods than if we had used separate random samples for each case.

It is also worthwhile looking at the finishing time of the runs that actually finished, shown in Fig. 5. As can be seen from the figure the hardness based fitness measure is significantly faster at finishing than the standard fitness. Thus not only is the hardness measure better at finding solutions, it is also finds those solutions faster.

Finally, we mention the computational resources required for our new method. There is no extra storage cost incurred using this implementation. Further, although the hardness computation is not optimized, its use adds only about 13% to the CPU time to evaluate each generation on a SUN SparcStation 2. This modest increase is more than offset by the large reduction in the problem solving steps, from an average of 49±1 generations using the standard fitness measure to 28±1 using the hardness-based measure (where the ± is the standard error of the mean), so that

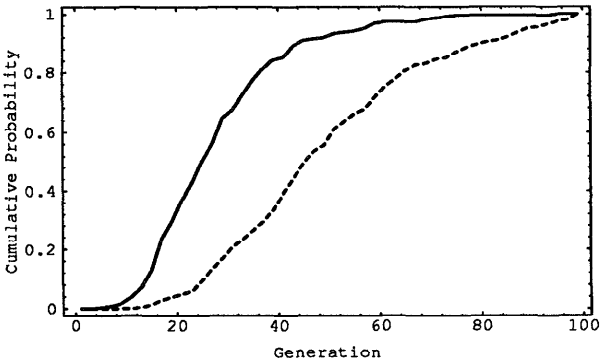

Fig. 5. Probability of finishing as a function of generation averaged over 400 runs. For this plot μ = 25 and k = 4, n_0 = 150. The solid line the hardness-based fitness results (224 finishers) and the dotted line is the standard fitness results (201 finishers). The chance that the two distributions are the same is rejected at a significance level of a few parts per million using the Kolmogorov-Smirnov test [Press et al., 1986].

our method uses 1.75 times fewer generations, on average, and thus runs 1.52 times faster in CPU time.

Diversity

One important characteristic of GAs is the diversity of the individuals in the population. The mutation rate is a force acting to increase diversity and the crossover operation is a force acting to reduce diversity. Diversity can be a good or bad depending on when it occurs during problem solving. During the early part of a search we would like to have a high diversity so as to reap the benefits of having many individuals exploring different parts of the search space. On the other hand, once a promising region of the space has been found we would like to focus our resources on that area which necessitates a reduction of the diversity.

We defined the diversity of the GA as the average over the population of the fraction of the least prominent value for a particular bit location in the state, averaged over all the locations. Thus, the maximum diversity of 0.5 means that 0's and 1's are evenly distributed, and the minimum diversity of 0 means that each member of the population is identical at each location. The evolution of diversity is shown in Fig. 6. As seen in the figure the diversity for the hardness-based runs is lower than for the standard runs in the region where the hardness-based runs typically finish. This means that the hardness switchover that occurs at maximum theoretical hardness is having the desired effect of focussing the individuals into easier regions of the problem and thereby speeding problem solving.

Extensions

In this paper we have shown how the simple use of a theoretically motivated measure of problem difficulty provides a useful heuristic for evaluating a population of search states. Within this context there are a variety of extensions that could also be tried. For instance, instead of focusing on how hard the remaining part of the problem

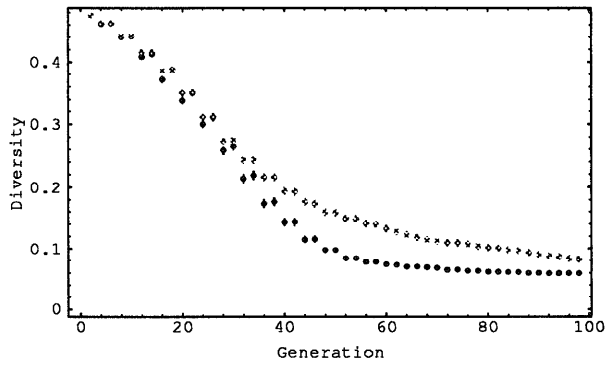

Fig. 6. Diversity as a function of generation for 400 runs. For this plot $\mu = 25$ and $k = 4$, $n_0 = 150$. The black points are for the hardness-based fitness result and the gray points are for the standard fitness result.

is to solve (favoring easier cases), we could also examine how hard a subproblem has already been solved. There are also more sophisticated ways to define the subproblem hardness. In particular, one could vary not only the number of constraints as we did, but also the number of variables. That is, only count those variables that are involved in at least one conflicting constraint. This would focus more precisely on the remaining subproblem, but ignores any interaction with the remaining constraints, which are already satisfied by the given state. If these remaining constraints greatly restrict the allowed choices for solving the subproblem, that subproblem will in fact be much more difficult than it appears on its own. This can be addressed to some extent by including the overlap of the remaining constraints with the subproblem as additional constraints for the subproblem. Including this level of detail requires using an extension of the basic theory that applies to minimized nogoods of differing sizes [Williams and Hogg, 1992a].

The idea of using general statistical knowledge of the problem may be extended to other types of problem solving techniques. For example, we may be able to use the hardness criteria to better adjust the cooling schedule in simulated annealing [Kirkpatrick et al., 1983]. Neural network learning may also benefit by using a hardness measure to automatically adjust the learning rate of the network as it learns. Heuristic search, such as beam search or heuristic repair [Minton et al., 1990] may also benefit.

A more intriguing possibility exploits the diversity behavior we found using the "stampede" across the critical region. In nature, this kind of high diversity to low diversity activity is seen with ants who initially forage over a wide area but become highly localized once a food source has been found. Similarly, one could imagine a collection of mobile robots [Brooks, 1991] trying to solve some kind of constraint satisfaction problem. Initially the robots would be highly dispersed but would become much more clustered as one of them informed the others that it had found something interesting to investigate. This would correspond to the genetic stampede we have observed.

References

Brooks, R. A. (1991). New approaches to robotics. *Science*, 253:1227–1232.

Cheeseman, P., Kanefsky, B., and Taylor, W. M. (1991). Where the really hard problems are. In Mylopoulos, J. and Reiter, R., editors, *Proceedings of IJCAI91*, pages 331–337, San Mateo, CA. Morgan Kaufmann.

Forrest, S. (1993). Genetic algorithms: Principles of natural selection applied to computation. *Science*, 261:872–878.

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

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

Grefenstette, J. (1990). A User's Guide to GENESIS Version 5.0. *Navy Center for Applied Research in AI, Naval Research Laboratory.*

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.

Johnson, R. A. and Wichern, D. W. (1992). *Applied Multivariate Statistical Analysis*. Prentice Hall, Englewood Cliffs, NJ, 3rd edition.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220:671–680.

Minton, S., Johnston, M. D., Philips, A. B., and Laird, P. (1990). Solving large-scale constraint satisfaction and scheduling problems using a heursitic repair method. In *Proceedings of AAAI-90*, pages 17–24, Menlo Park, CA. AAAI Press.

Mitchell, D., Selman, B., and Levesque, H. (1992). Hard and easy distributions of SAT problems. In *Proc. of 10th Natl. Conf. on Artificial Intelligence (AAAI92)*, pages 459–465, Menlo Park. AAAI Press.

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. (1986). *Numerical Recipes*. Cambridge Univ. Press, Cambridge.

Williams, C. P. and Hogg, T. (1992a). Exploiting the deep structure of constraint problems. Technical Report SSL92-24, Xerox PARC, Palo Alto, CA.

Williams, C. P. and Hogg, T. (1992b). Using deep structure to locate hard problems. In *Proc. of 10th Natl. Conf. on Artificial Intelligence (AAAI92)*, pages 472–477, Menlo Park, CA. AAAI Press.