Genetic Algorithm Behavior in the MAXSAT Domain

Soraya Rana and Darrell Whitley

Colorado State University, Fort Collins CO. 80523, USA
{rana, whitley}@cs.colostate.edu

Abstract. Random Boolean Satisfiability function generators have recently been proposed as tools for studying genetic algorithm behavior. Yet MAXSAT problems exhibit extremely limited epistasis. Furthermore, all nonzero Walsh coefficients can be computed *exactly* for MAXSAT problems in polynomial time using only the clause information. This means the low order schema averages can be computed quickly and exactly for very large MAXSAT problems. But unless P=NP, this low order information cannot reliably lead to the global optimum, thus nontrivial MAXSAT problems must be deceptive.

1 Introduction

A common assumption is that a simple genetic algorithm assembles a solution using low-order schema information. When an optimization problem has low-order schema information that leads away from the global optimum, it is considered **deceptive**. In this paper we examine MAXSAT as an example of a problem domain that can be proven to contain misleading low order schema information. Furthermore, MAXSAT search spaces tend to result in similar schema fitness averages. These two attributes of MAXSAT problems make them an unsuitable application for traditional genetic algorithms.

Genetic algorithms are typically applied to black-box optimization problems; however, MAXSAT problems are not black-box optimization problems. As it turns out, the problem description for arbitrary random MAXSAT problems can be used to compute low order schema information **exactly** and in a linear amount of time with respect to the number of clauses. If $P \neq NP$, then this low order information cannot reliably lead to a global optimum. We will empirically illustrate that a genetic algorithm converges to many distant points and that those points are only partially consistent with the low order schema information.

2 A Walsh Analysis of Satisfiability Problems

A method for studying the epistasis in a binary function is to use Walsh analysis [5, 6, 9]. All binary functions can be represented as a weighted sum of **Walsh**

[0] Copyright ©1998, Springer-Verlag. http://www.springer.de/comp/lncs/index.html

functions denoted by ψ_j, where $0 \leq j \leq 2^L - 1$ with each Walsh function being $\psi_j : \mathcal{B}^L \rightarrow \{-1, 1\}$. The real-valued weights are called **Walsh coefficients**. Walsh functions are defined using a bitwise-AND of the function index and its argument. Note that operations on indices such as j act on the binary representation of j. Let $bc(j)$ be a count of the number of 1 bits in string j. The 2^L Walsh coefficients for function $f(i)$ can be computed by a Walsh transform:

$$w_j = \frac{1}{2^L} \sum_{i=0}^{2^L-1} f(i)\psi_j(i) \quad \text{where} \quad \psi_j(x) = (-1)^{bc(j \wedge x)}$$

So, if $bc(j \wedge x)$ is odd, then $\psi_j(x) = -1$ and if $bc(j \wedge x)$ is even, then $\psi_j(x) = 1$. Note these equations require that the enumeration and evaluation of all 2^L strings to compute a single coefficient. Thus, Walsh analysis generally requires exponential time with respect to L.

The Boolean Satisfiability problem (SAT) is to determine whether or not there is some setting of variables such that a Boolean expression can be made TRUE. SAT problems are generally presented in conjunctive normal form using clauses of a particular length, K. When KSAT problems are solved using optimization algorithms rather than backtracking algorithms, all variables are given a setting and evaluated. However, black box optimizers cannot effectively optimize a function that returns nothing but a 0 or 1. Rather than combining the clauses using the AND operator, the numeric truth value for each individual clause, 0 for FALSE or 1 for TRUE, are summed together. This form of the evaluation function is called MAXSAT. An L-bit MAXSAT problem can be represented as a sum of C disjunctive clauses, f_i:

$$f(x) = \sum_{i=1}^{C} f_i(x)$$

where $f, f_1, f_2, ...f_C : \mathcal{B}^L \rightarrow \mathcal{R}$. Each clause evaluation, f_i, takes an L-bit string as input but extracts and uses only K-bits in the calculation, where K is the length of the clause. This constraint means that each clause contributes to only 2^K Walsh coefficients [8].

Since the Walsh transform can be performed by a simple linear transformation, the Walsh transform of a MAXSAT problem can be treated as a sum of the Walsh transforms of the individual clauses.

$$W(f(x)) = \sum_{i=1}^{C} W(f_i(x))$$

Consider a 3-bit single clause problem using $f(x) = \neg x_2 \vee x_1 \vee x_0$. We construct the Walsh matrix to compute the Walsh coefficient vector, w, using matrix

multiplication.

$$\boldsymbol{w} = \frac{1}{8}\begin{bmatrix}1\\1\\1\\1\\0\\1\\1\\1\end{bmatrix}^T \begin{bmatrix}1&1&1&1&1&1&1&1\\1&-1&1&-1&1&-1&1&-1\\1&1&-1&-1&1&1&-1&-1\\1&-1&-1&1&1&-1&-1&1\\1&1&1&1&-1&-1&-1&-1\\1&-1&1&-1&-1&1&-1&1\\1&1&-1&-1&-1&-1&1&1\\1&-1&-1&1&-1&1&1&-1\end{bmatrix} = \begin{bmatrix}0.875\\-0.125\\-0.125\\-0.125\\0.125\\0.125\\0.125\\0.125\end{bmatrix} \quad (1)$$

The fitness vector for any disjunctive clause must be 1 everywhere except for a single 0 where the clause is FALSE. Since ψ_0 is the average fitness, $w_0 = \frac{2^K-1}{2^K}$. The remaining Walsh functions evaluate to -1 exactly as often as 1. All but one value will cancel and this value depends on where the fitness is zero. Since the clause is disjunctive, the only time the fitness is zero is when all literals are simultaneously FALSE. Define the function $z = negvec(c, L)$ to take a clause description as input and return an L-bit string that makes the clause false. Variables included in the vector that are unused in the clause will be filled with 0's in z. The remaining Walsh coefficients can be written as:

$$w_j = -\frac{1}{2^K}\psi_j(negvec(c, L)) \qquad \forall j \neq 0$$

3 Observations about the Walsh Analysis of MAXSAT

The number of Walsh coefficients is exponential with respect to K; however, $K << L$ and is typically a bounded constant. The most commonly used value for K is 3 which will be adopted for the remainder of the paper. There are only $7C$ calculations required to enumerate the set of nonzero Walsh coefficients for arbitrary MAX3SAT expressions. Normally the value of C is linear with respect to L; therefore, the set of nonzero Walsh coefficients is sparse and enumerable in polynomial time. Furthermore, the set of all nonzero Walsh coefficients represent nothing but simple counts of variable uses over the set of clauses.

Consider a small MAXSAT function $f : \mathcal{B}^4 \rightarrow \mathcal{R}$ with $f(x) = f_1 + f_2 + f_3$ and

$$f_1 = (\neg x_2 \vee x_1 \vee x_0) \qquad f_2 = (\neg x_3 \vee x_2 \vee x_1) \qquad f_3 = (x_3 \vee \neg x_1 \vee \neg x_0).$$

The Walsh coefficient w_2 is computed for f as follows:

$$w_2 = -\frac{1}{8}\psi_2(negvec(f_1, 4)) - \frac{1}{8}\psi_2(negvec(f_2, 4)) - \frac{1}{8}\psi_2(negvec(f_3, 4))$$

The w_2 coefficient is a measure of the linear contribution of the Boolean variable x_1. All three clauses use the variable x_1 so they are all included in the calculation.

The Walsh function indices are masks that isolate variables or combinations of variables. The $negvec()$ function produces a mask corresponding to negated variables. When this vector and the Walsh function index are merged together by

the bitwise-AND, the information that is extracted is the parity of the negation information for a particular subset of variables.

Thus, the Walsh coefficients for MAXSAT problems are nothing more than a constant $-\frac{1}{2^K}$ multiplied by a count over the uses of subsets of variables across the set of all clauses. The following method shows how to compute a simple count that can be used to calculate the nonzero Walsh coefficients for arbitrary MAXSAT problems.

Define $T(x, c)$ for variable x and clause c as follows:

$$T(x,c) = \begin{cases} 1 \text{ if } x \text{ is present} \\ -1 \text{ if } \neg x \text{ is present} \\ 0 \text{ otherwise} \end{cases}$$

Define $use(c)$ to return a subset of all variables used by a clause c. Let $S_{v,c}$ be a sign (± 1) determined for a clause c and a subset of variables v as follows:

$$S_{v,c} = \prod_{x \in v} T(x,c) \quad \text{where} \quad v \in \bigcup_{c \in f} (\mathcal{P}(use(c))$$

Then the Walsh term associated with a particular variable combination v is

$$w_v = \frac{-1}{2^K} count_v \quad \text{where} \quad count_v = \sum_c S_{v,c}$$

For order-1 Walsh coefficients, these counts simply compute the difference between the number of times a variable is used positively and the number of times a variable is used negatively.

Schema averages are sometimes used to try to understand the behavior of genetic algorithms. Order-1 schema are computed using Walsh coefficients w_0 and w_i, where w_i measures the contribution of a single bit [5]. The order-1 schema averages are only a constant offset of the order-1 Walsh coefficients. Thus, all order-1 schema competitions are *decided* by the following heuristic:

> If a variable occurs positively more often than negatively, the variable should be set to TRUE otherwise set the variable FALSE.

It follows that if the order-1 schema are not misleading, then this heuristic *decides* the MAXSAT problem. In practice, all non-trivial MAXSAT problems are *deceptive:* the schema information is inherently misleading. We might also ask if there is useful order-2 or order-3 schema information. Figure 1 shows the average magnitude of *nonzero* Walsh coefficients for a large set of randomly generated 100 variable MAX3SAT problems. The horizontal axis represents the ratio of clauses to variables; steps were taken in increments of 0.2. Each point in the graph is the average of 30 problem instances. Each line tracks the average magnitude of all nonzero Walsh coefficients for order-1, order-2, and order-3 interactions.

The plot indicates that the order-1 terms are relatively large compared to the order-2 and order-3 Walsh coefficients. Since order-2 and order-3 schema

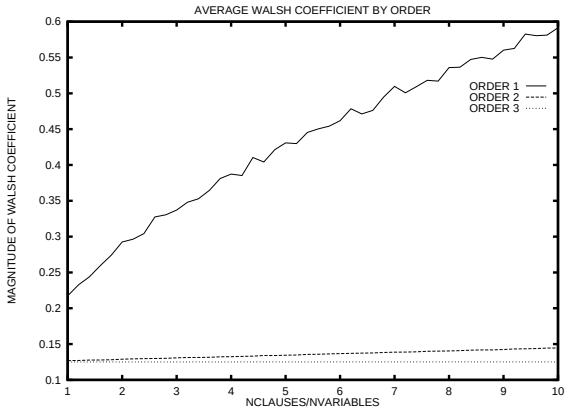

Fig. 1. Average magnitudes of nonzero Walsh coefficients (excluding w_0).

averages include order-1 Walsh coefficients, it follows that these schema averages will also tend to be consistent with the order-1 schema averages. Furthermore, since all of this information can be computed in polynomial time and it is fairly consistent, this information cannot lead to the global optimum if $P \neq NP$. Therefore, MAXSAT problems must be deceptive to a genetic algorithm. Also notice that many of the Walsh coefficients are the same (either 0 or 0.125), and thus many schema averages will be identical. Not only is the MAXSAT landscape misleading, it is also relatively flat.

4 Empirical Verification

We have provided theoretical evidence to support the claim that genetic algorithms should not perform well in the MAXSAT domain. To verify this, we ran a Simple Genetic Algorithm (SGA) with tournament selection on a small set of random MAXSAT problem instances. Our SGA implementation used tournament selection with a tournament size of 4, population size of 500, mutation rate 0.1, and probability of crossover of 0.6. The purpose of this experiment is to analyze the convergence points of the genetic algorithm. We ran two other algorithms to illustrate that our example problems are reasonable MAXSAT problems. The first algorithm is the Davis-Putnam (DP) algorithm [2], a well known complete algorithm for solving SAT problems. The second algorithm is a stochastic greedy hill-climbing search algorithm known as GSAT [10].

All three algorithms were designed using different notions of work. For Davis-Putnam, work is measured by the number of recursive calls (i.e. size of the search tree). The measure of work for GSAT is the number of flips taken (i.e. path length). Since the genetic algorithm is a black box optimizer, the measure of work for the SGA is the number of evaluations. Although the measures of work differ, we gave all three algorithms a fair chance to solve the problem. The DP algorithm is deterministic so it was run only once on each of the test problems.

Prob.	SGA		DP	GSAT			
	Solved	Evals	Calls	Solved	Flips	Side	Down
Underconstrained	19	22250.9	32	20	160.2	136.5	23.7
Hard	2	39246.5	147	15	1762.73	2406.2	165.85
Overconstrained	8	32545.9	92	20	1001.2	898.7	102.5

Table 1. Results of running SGA, Davis-Putnam, GSAT and SGA on three MAX3SAT problem instances of varying levels of difficulty.

Since GSAT and the SGA will converge to potentially different locations, we ran both algorithms 20 times on each example problem. For each run, GSAT was given a rather stringent cutoff of 10 tries per run and 500 flips per try. The genetic algorithm was allowed to run for 200 generations, which was sufficient for the population to converge for every run.

As the ratio of clauses to variables is varied in MAX3SAT problems, a phase transition occurs where random problem instances transition from being generally SAT to generally UNSAT. The problem difficulty has an easy-hard-easy pattern with the hard problems falling in the phase transition. This transition occurs when the number of clauses is approximately 4.3 times the number of variables [1, 7]. We randomly generated three 100-variable MAX3SAT problems for this small experiment. The problems were generated with respect to a target string so they were guaranteed to be satisfiable. The three example problems were randomly generated using 300 clauses (underconstrained), 430 clauses (hard) and 600 clauses (overconstrained).

Table 1 lists the results of running all three algorithms on the example problems. The genetic algorithm located solutions to all three problems but solved the hard problem very infrequently. When an incomplete search algorithm such as a genetic algorithm is run on SAT problems, if it is unable to locate a solution reliably, then it will mistakenly indicate that the problem is UNSAT. The genetic algorithm also ran several orders of magnitude slower than either DP or GSAT. Davis-Putnam solved all three problems very quickly with relatively few recursive calls. GSAT also solved all three problems, although it was unable to locate an optimum on all 20 runs on the hard problem. The average number of flips (steps) taken by GSAT for the solved problem instances is also given. The behavior of all of the algorithms corresponds to the familiar easy-hard-easy pattern for the three problems.

The GSAT experiment provides some very useful information about the structure of the MAXSAT search space. The MAXSAT search space is known to be made up of many flat plateaus [4]. The GSAT experiments illustrate the amount of time spent wandering through flat regions. Two extra columns are listed in the table for GSAT, *Side* and *Down*. GSAT is a steepest descent bit-climber that randomly chooses between the set of best available moves. GSAT will always take moves until it has reached the designated $maxflips$. The first column *Side* lists the average number of sideways moves (i.e. where the best available move does not change the evaluation) taken on a single run. The second

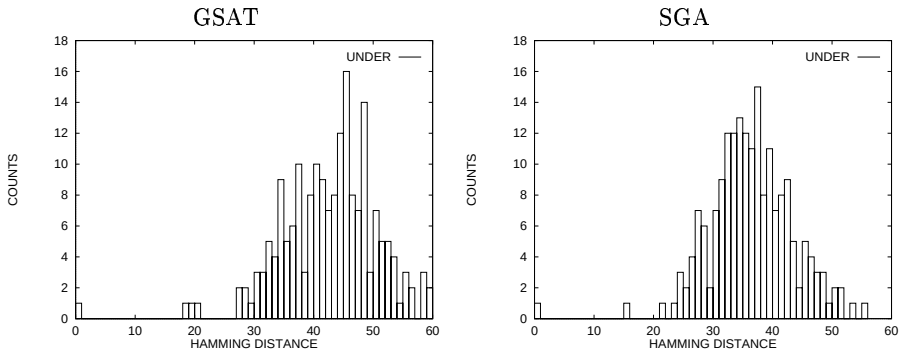

Fig. 2. Distance between convergence points on the underconstrained problem.

column, *Down*, corresponds to improving moves since our evaluation function tracks the number of UNSAT clauses. No uphill (worse) moves were needed to solve these problems. Clearly, GSAT spends the vast majority of time wandering across flat plateaus. Without some mechanism for moving across plateaus, any search algorithm would be at a serious disadvantage.

The analog to plateaus in the context of schemata are schema averages that are similar (or identical). Considering the limited set of Walsh coefficients, it would appear that many schemata should share identical fitnesses. In other words, just as the space appears flat to GSAT, the space will also appear flat to a genetic algorithm. However, a traditional genetic algorithm has no way to deal with this problem.

4.1 Distances between Convergence Points

A set of convergence points for SGA and GSAT was generated from the 20 different runs on each example problem. SGA would often converge to a set of slightly different points that shared the same best fitness. When there were ties, the convergence point was created using votes for each bit value by the best individuals. The distances between convergence points can indicate whether or not there are large regions of the space that SGA and GSAT find particularly attractive. If there are specific regions that are good, then the convergence points should be close together. Otherwise, the convergence points will be far apart. For each test problem, we computed the Hamming distances between all pairs of convergence points and created a histogram to indicate how those distances were distributed. There were 20 convergence points and thus 190 pairings of convergence points.

Figure 2 shows the distribution of distances between the set of convergence points for GSAT and SGA on the underconstrained problem. Recall that SGA solved this problem 19 times and GSAT solved this problem 20 times. Thus, most of the convergence points represent a satisfying solution. The Hamming distance between solutions, however, is very large. The distances are, in fact, consistent

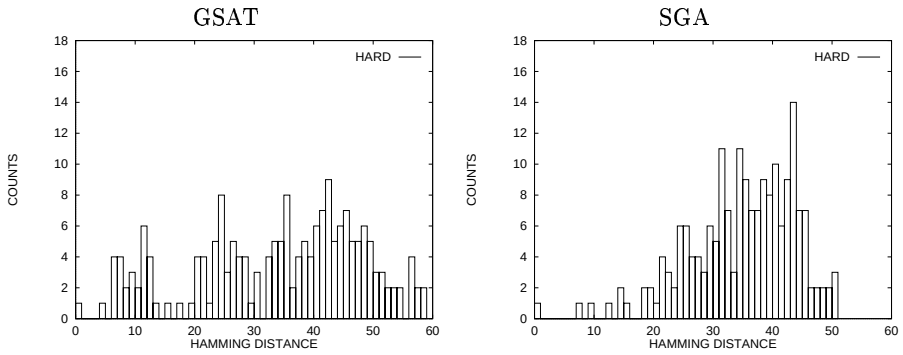

Fig. 3. Distance between convergence points on the hard problem.

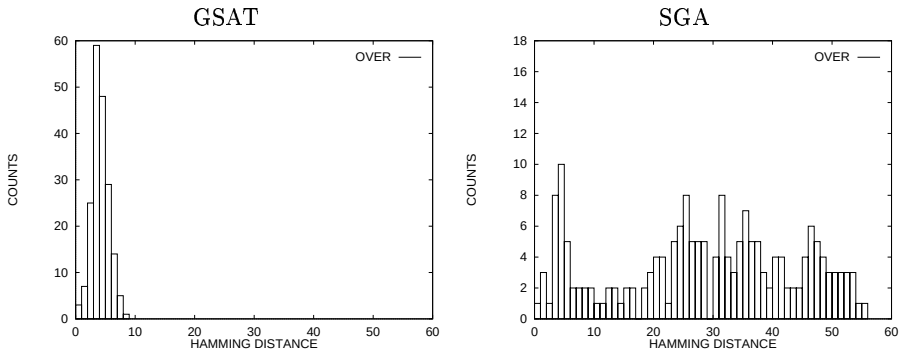

Fig. 4. Distance between convergence points on the overconstrained problem.

with randomly generated pairs of strings. Thus, there are many attractive regions that contain global optima. In some sense, this problem is easy because there are solutions scattered throughout the search space.

The histograms in Figure 3 present the Hamming distances between the convergence points for the hard problems. The distribution of Hamming distances for GSAT tend to be flatter than the SGA. The GSAT histogram also appears to be multimodal. This indicates that many solutions were clustered together but the clusters were relatively distant. This is indicative of there being a few distinct regions that contain global optima. However, given the amount of exploration done by GSAT and the poor performance of the genetic algorithm, it also appears that there are many highly fit regions that do not contain global optima.

The last figure, Figure 4 presents the distributions of Hamming distances between convergence points on the overconstrained problem. Notice that the histogram for GSAT required a larger range than all other histograms along the y-axis. It is clear that the convergence points for GSAT were all clustered together in one large region. The SGA Hamming distance distribution is multi-

		L-String			Global Target		
		All	Global	Local	All	Global	Local
UNDER	GSAT	36.45	36.45	–	45.25	45.25	–
	SGA	29.95	30.32	23.00	44.85	44.95	43.00
HARD	GSAT	36.45	37.13	34.40	35.45	36.80	31.40
	SGA	32.70	35.00	32.44	40.50	34.00	41.22
OVER	GSAT	33.30	33.30	–	4.1	4.1	–
	SGA	32.40	32.75	32.17	20.60	4.75	31.17

Table 2. Distances between the convergence points and the low order string and a known optimum.

modal. The peak that occurs in the low range of Hamming distances corresponds to the differences between convergence points that were global optima. The local optima form the second set of larger Hamming distances.

The Hamming distances between convergence points indicates that the SGA and GSAT were converging to points that were very different from one another. This behavior is consistent with the argument that the MAXSAT landscape is very flat. Essentially, there is little information to guide the search in any predictable way. The next question to ask is whether or not the SGA convergence points were consistent with the low order schema information.

4.2 Where the Low Order Schemata Lead ...

Deception occurs when low order schema information does not lead to a global optimum. To examine whether or not this occurs in MAXSAT, we defined the **L-string** to indicate the string that is most consistent with the low order schema information. To construct this string, we enumerated all possible order-3 partitions and ranked the schemata. All top ranked schemata contributed votes for its fixed bit positions. The order-1 information was not used to generate the **L-String** because approximately $10-15$ bit positions had Walsh coefficients of 0 and would result in a partially specified string. The remaining positions that were specified by order-1 schema information were identical to the corresponding values in the **L-String**. In all cases, the **L-String** was not a global optimum.

The **L-String** and a second string, the **Global Target**, were compared to each of the convergence points. The **Global Target** strings were the strings used to create each of the random problem instances. Table 2 lists the average distances distances between the convergence points and the **L-String** and **Global Target**. The column **All** lists the difference between the L-String and Global Target compared to the set of all convergence points. The two other columns, **Global** and **Local** represent averages over the subsets of convergence points that were global optima and local optima respectively. The purpose for partitioning the set of convergence points is to determine whether or not the global optima tend to be clustered together. The L-Strings tend to differ from all convergence points by more than 30 bits indicating inconsistencies between low order and high order schema information. The Global Target string is also far from the convergence points for the underconstrained and the hard problems. However,

it is obvious that the global optima are tightly clustered into one region in the overconstrained problem.

5 Conclusions

This paper has reviewed theoretical observations about the MAXSAT domain and the inevitability of deception in this domain. We have supported the theory by studying the convergence points of a Simple Genetic Algorithm. For three MAXSAT examples of varying difficulty, the low order information differed by more than 30% from the convergence points. Furthermore, the convergence points were not localized. The convergence points also tended to differ from one another by 30%. This behavior indicates that the problems are not only deceptive, but there are also many equally good regions for a genetic algorithm to explore. If there were biases in the schemata, the genetic algorithm would have converged consistently to specific regions of the search space for the same problem. Yet the convergence behavior appears to be almost random.

Larger empirical studies of genetic algorithms for optimization of MAXSAT problems have illustrated that a traditional genetic algorithm does not perform well [3]. The theoretical results and this small set of results illustrate that algorithms that solely use schema information to search will be at a disadvantage in the MAXSAT domain.

References

1. P. Cheeseman, B. Kanefsky, and W. M. Taylor. Where the *really* hard problems are. In *Proc. Twelfth International Joint Conference on Artificial Intelligence*, 1991.
2. M. Davis and H Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7:201–215, 1960.
3. J. Frank. A Study of Genetic Algorithms to Find Approximate Solutions to Hard 3CNF Problems. *Golden West International Conference on Artificial Intelligence*, 1994.
4. J. Frank, P. Cheeseman, and J. Stutz. When gravity fails: Local search topology. *Journal of Artificial Intelligence Research*, 7:249–281, 1997.
5. D. Goldberg. Genetic algorithms and walsh functions: Part I, a gentle introduction. *Complex Systems*, 3:129–152, 1989.
6. D. Goldberg. Genetic algorithms and walsh functions: Part II, deception and its analysis. *Complex Systems*, 3:153–171, 1989.
7. D. Mitchell, B. Selman, and H. Levesque. Hard and easy distribution of sat problems. In *Proc. Tenth National Conference on Artificial Intelligence*, San Jose, CA, 1992.
8. S. Rana, R. Heckendorn, and D. Whitley. A tractable walsh analysis of Sat and its implications for genetic algorithms. In *Proc. Fifteenth National Conference on Artificial Intelligence*, 1998.
9. C. Reeves and C. Wright. Epistasis in genetic algorithms: An experimental design perspective. In Larry Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 217–224. Morgan Kaufmann, 1995.
10. B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proc. Tenth National Conference on Artificial Intelligence*, pages 440–446, San Jose, CA, 1992.