The Performance of Genetic Algorithms on Walsh Polynomials: Some Anomalous Results and Their Explanation

Stephanie Forrest
Melanie Mitchell

SFI WORKING PAPER: 1991-05-024

SANTA FE INSTITUTE

The Performance of Genetic Algorithms on Walsh Polynomials: Some Anomalous Results and Their Explanation

Stephanie Forrest
Dept. of Computer Science
University of New Mexico
Albuquerque, N.M. 87131-1386

Melanie Mitchell
Artificial Intelligence Laboratory
University of Michigan
Ann Arbor, MI 48109-2110

Abstract

In this paper we discuss a number of seemingly anomalous results reported by Tanese concerning the performance of the genetic algorithm (GA) on a subclass of Walsh polynomials. Tanese found that the GA optimized these functions poorly, that a partitioning of a single large population into a number of smaller independent populations seemed to improve performance, and that hillclimbing outperformed both the original and partitioned forms of the GA on optimizing these functions. We reexamine these results experimentally and theoretically, and propose and evaluate some explanations. In addition, we examine the question of what are reasonable and appropriate ways to measure the performance of genetic algorithms.

1 Introduction

Tanese's 1989 doctoral dissertation reports experiments that apparently contradict some commonly held beliefs about genetic algorithms (GAs) as function optimizers [13, 12]. Specifically, Tanese's results show that for a certain class of Walsh polynomials, GAs are poor optimizers, perform worse than hillclimbing, and often perform best when the total population is split up into very small subpopulations. These results call into question several expectations about GAs — that they are good at optimizing Walsh polynomials [1], that they will routinely outperform hillclimbing and other gradient descent methods on hard problems such as those with nonlinear interactions [9], and that populations must be of a sufficient size to support effective schema processing [4].

Tanese's results are provocative and have caused some researchers to question the effectiveness of GAs in function optimization. Thus, they deserve a careful explanation. This paper reviews these apparently negative results, explains some of the anomalies, and discusses the general question of what criteria are appropriate to use in assessing the performance of GAs.

2 Experimental Setup

The experiments we report in this paper were performed with a similar GA and identical parameter values to those used by Tanese [2, 13]. All of Tanese's experiments used strings of length 32 and populations of 256 individuals. The population was sometimes subdivided into a number of smaller subpopulations. Tanese's algorithm used a *sigma scaling* method, in which the number of expected offspring allocated to each individual is a function of the individual's fitness, the mean fitness of the population, and the standard deviation from the mean. An individual with fitness one standard deviation above the population mean was allocated one and a half expected offspring. Multipoint crossover was used, with a crossover rate of 0.022 per bit (e.g., for 32-bit strings, there were on average 0.7 crossovers per pair of parents per generation). The crossover rate (per pair) was interpreted as the mean of a Poisson distribution from which the actual number of crosses was determined for each individual. The mutation probability was 0.005 per bit. With the exceptions of sigma scaling and multipoint crossover, Tanese's GA was conventional. Tanese ran each of her experiments for 500 generations, whereas we ran each of ours for 200 generations, although, as will be discussed later, this was not an important difference. For some experiments we altered certain parameter values; these exceptions are noted explicitly.

3 Walsh Polynomials

In her dissertation, Tanese describes experiments involving several fitness functions. In this paper we consider only the results with respect to Walsh polynomials. Walsh polynomials and their relevance to

GAs have been discussed by a number of researchers [1, 5, 9, 13]. A Walsh polynomial is defined as a sum of *Walsh functions.*

There is a correspondence between certain partionings of the GA's search space and Walsh functions. Such partitionings are induced by schemas. For example, the partitioning d**...* divides the search space into two halves, corresponding to the schemas 1**...* and 0**...*. Likewise, the partitioning dd**...* represents a division of the space into four quarters, each of which corresponds to a schema with the leftmost two bits defined. Each different partitioning of the search space can be indexed by a unique binary integer (bit string) in which 1's correspond to the partition's defined bits and 0's correspond to the non-defined bits. For example, under this enumeration, the partition d**...* has index j=100...0. The Walsh function corresponding to the jth partition is defined as follows (where both j and x are bit strings):

$$\psi_j(x) = \begin{cases} 1 & \text{if } x \wedge j \text{ has even parity} \\ -1 & \text{otherwise.} \end{cases}$$

A Walsh polynomial has the following form:

$$f(x) = \sum_{j=0}^{2^l-1} \omega_j \psi_j(x)$$

where l is the length of the bit string x, and each ω_j is a real-valued coefficient. Walsh polynomials provide a basis for defining any real-valued function on bit strings.

From this general class of functions, specific subsets were selected as fitness functions, both for Tanese's work and for the experiments reported here. Tanese generated each fitness function, hereafter called *Tanese functions*, by randomly choosing 32 partition indices j (called the *defined partitions*), all of the same order — that is, all containing the same number of 1's. The coefficient ω_j for each of the 32 chosen partition indices was also chosen at random from the interval $(0.0, 5.0]$. The fitness function consisted of the sum of these 32 terms (all other coefficients were effectively set to 0).

Once the 32 partition indices were chosen, a point x' was chosen randomly to be the global optimum, and the sign of each of the 32 ω_j's was adjusted so that the fitness of x' would be $\sum |\omega_j|$.

This method of constructing functions had several advantages for comparing the performance of different GAs: (1) it was easy to construct random functions that on average were of similar difficulty (although Goldberg [6] has shown that it is possible to construct two functions of a given order that have different levels of difficulty for the GA); (2) functions of different degrees of difficulty could be constructed by varying the order, since low-order functions of this sort should on average be easier for the GA to optimize than high-order functions [13]; and (3) the global optimum was known, which made it possible to determine how close the GA came to the maximum value of the function.

Tanese conducted experiments on fitness functions with defined partitions at orders 4, 8, 16, and 20 (each function had all of its defined partitions at the same order). For each experiment she randomly generated 64 functions of a given order, and compared the performance of the *traditional* (single population) GA with a number of *partitioned* GA's, in which the population of 256 individuals was subdivided into various numbers of smaller populations. In this paper, we discuss results only for functions at order 8, since these were the functions Tanese analyzed in the greatest depth. All of our experiments involved manipulations of parameters on the traditional GA; we did not experiment with partitioned GA's. For each of our experiments, we ran the GA once on each of 20 different randomly generated functions, for 200 generations each. Tanese carried each run out to 500 generations, but in each of our runs that used strings of length 32, the population had reached a more or less steady state by about generation 100, and the results would not have changed significantly if the run had been extended. The shorter runs were sufficient for determining the comparative effects of the various manipulations we performed on the parameters.

4 Discussion of Anomalies

This section discusses and proposes explanations for the following anomalies in Tanese's results :

- The poor performance of all forms of the GA on optimizing both low-order and high-order Tanese functions.
- The improvement in optimization performance obtained by the partitioned GA over the traditional GA.
- The superior optimization performance of hill-climbing on the Tanese functions as compared with both the traditional and partitioned GA.

4.1 Why does the GA optimize Walsh polynomials poorly?

One of Tanese's most striking results was the poor performance of the GA (in both its traditional and partitioned-population form) when searching for maximum values of the functions described above. In Tanese's experiments, the GA was run 5 times on each of the 64 randomly generated functions; each set of 5 constitutes a *trial.* One of her primary criteria was the *success rate*: the number of trials on which the global optimum was found at least one out of five times. On the 64 trials (i.e., 320 runs total) on randomly generated order 4 Walsh polynomials, the success rate of the traditional GA was only 3 (the most successful partitioned algorithm's success rate was only 15). On 320 runs on randomly generated order 8 Walsh polynomials, neither the traditional nor the various partitioned GAs ever found the optimum.

The following possible explanations for the GA's poor performance will be discussed and evaluated in this section: (1) the average defining-lengths of schemas are very long and thus good schemas tend to be broken up by crossover; (2) randomly generating 32 partition-indices over strings of length 32 results in a large number of overlaps, which means that most of the significant loci are correlated, effectively making the functions very difficult; (3) crossover is ineffective on these functions because of the lack of lower-order building blocks; and (4) the fitness functions are deceptive.

4.1.1 Is the GA's poor performance due to long defining lengths of schemas?

Given a Tanese function F, the fitness of a string x under F depends on the parity of $x \wedge j$ for each j in the terms of F. Because of this parity calculation, a change of a single bit in x in any of the positions in which j has a 1 will produce an opposite value for $x \wedge j$ and thus reverse the contribution of that term to the total fitness. This implies that in general, for a Tanese function of order n, no schema s of order less than n will give the GA any useful information, since for any given j, half the instances of s will have even parity with respect to j, and half will have odd parity. This property is due to the parity properties of the Walsh functions and to the fact that in a Tanese function, all the terms are of the same order. This property and its implication for the effectiveness of crossover on these functions will be discussed in Section 4.1.3.

The expected defining length of a schema of order 8 in a 32-bit string is 26, a substantial proportion of the entire string [7]. Such long defining lengths could make the Tanese functions hard for the GA because of the high probability of crossover disruption, To what degree was this problem responsible for the poor performance of the GA on these functions?

To answer this question, we ran the traditional GA with Tanese's parameters on 20 randomly generated functions of order 8 whose partition indices were restricted to have a maximum defining length of 10. That is, for each of the 32 partition indices, a randomly positioned window of 10 contiguous loci was chosen, and all eight 1's were placed randomly inside this window.

Using the success-rate criterion described above, the performance was identical to Tanese's original results: the traditional GA never found the optimum. Other performance measures made it clear that limiting the defining length improved the GA's performance to some extent: the GA was able to find strings with slightly higher fitnesses (in terms of percent of optimum) and slightly higher mean-population fitnesses. These results, together with results from all experiments described in this paper, are summarized in Table 1 under the heading *Def-Len: 10.* They are to be compared with the values under *Original*, giving the results from our replication of Tanese's traditional GA runs on order-8 functions. We conclude that the contribution of long defining lengths to the GA's overall poor performance is not significant. As will be discussed, one reason for this is that crossover is not very effective on these functions in the first place.

4.1.2 Is the GA's poor performance due to overlap among significant loci in the partition indices?

Next, we considered the possibility that overlaps among significant loci in the partition-indices j (i.e., loci with 1's) were causing correlations among terms in the fitness functions, making the optimization problem effectively higher-order. As a simple example, suppose that 0011 and 0110 are two order-2 j's that have non-zero positive coefficients. There are eight strings that will cause $\psi_{0011}(x)$ to be positive, corresponding to the schemas ##00 and ##11. Likewise, there are eight strings that will cause $\psi_{0110}(x)$ to be positive, corresponding to the schemas #00# and #11#. So, in order to find a point that gets a positive value from both $\psi_{0110}(x)$ and $\psi_{0011}(x)$, the GA must discover either the schema #000 or the schema #111. The net effect of this overlap is that three bits are correlated instead of two, making the problem effectively order-3 instead of order-2. In the case of the Tanese functions, this is a likely source of difficulty; for example, with order-8 functions, where 32 order-8 terms were randomly generated, each 1 in any given j will on average be a member of 7 other different j's, and thus the effective linkage will be exceedingly high.

To assess the effect of overlaps, we ran the GA on strings of length 128 rather than 32, in order to reduce the number of overlaps. With a string length of 128, each defined locus with allele 1 would participate on average in only 2 of the 32 partition indices. As before, we generated 20 random functions and ran the GA for 200 generations on each. As shown in Table 1 (under *Str-Len: 128*), the GA's performance was remarkably improved. The GA found the optimum 19/20 times, compared with 0/20 for the 32-bit case, and came very close to finding the optimum on the other run. Of all the experiments we tried, this caused the most dramatic improvement, leading us to conclude that the principle reason the Tanese functions are difficult is because the short strings (32 bits) and relatively high number of terms (32) causes nearly all 32 bits to be correlated, thus effectively creating an order 32-problem. In the non-overlapping case, it is possible for the GA to optimize each term of the function almost independently.

The fact that overlap is much higher with 32-bit functions than with 128-bit functions explains the strikingly different dynamics between runs of the GA on the former and latter functions. Space constraints prevent us from reporting these data in detail (see [3]), but we describe them qualitatively.

A Walsh polynomial can be thought of as a "constraint satisfaction" problem in which each term produces an additional constraint. The goal is to find an individual that receives positive values from as many terms as possible, and preferably from terms with high coefficients. In a typical run on 32-bit strings, the GA very quickly finds its highest-fit individual, by around generation 30. This individual receives positive values from only a subset of the terms in the fitness function, leaving the rest of the terms "unsatisfied," that is, yielding negative values on that individual. However, because of the high degree of overlap, the constraint satisfaction problem here is very severe, and to discover an individual that receives additional positive values from other terms — without losing too many of the currently held positive values — is very difficult (especially since crossover is largely ineffective on these functions). This is particularly true since, once individuals of relatively high fitness are discovered, the diversity of the population falls very quickly. For example, on one typical run the number of different chromosomes in the population started out at 256, fell to 39 by generation 20, and stayed relatively constant thereafter[1]. Very quickly, the population subdivides itself into a small number of mutually exclusive sets: one large set of individuals receiving positive values from one large subset of the terms in the fitness function, and other, much smaller sets of individuals receiving positive values from the other, smaller subsets of terms in the fitness function.

To summarize, the GA stays stuck at a local optimum that is discovered early. It is possible that raising crossover and mutation rates might improve the GA's performance on such functions, since it would increase the amount of diversity in the population.

On a typical run involving strings of length 128, the situation is very different. In such a run, the GA tends not to discover its highest-fit individual until late in the run (on average, around generation 150), and the diversity of the population remains relatively high throughout the run. The continuing high diversity of the population seems to be a result of several factors, including the following: since the problem is less constrained, there are many ways in which an individual can achieve a relatively high fitness; and since the crossover rate was defined on a per-allele basis, there are on average a greater number of crossovers per chromosome here than in the 32-bit case. However, the relative lack of constraints was the major factor, since when crossover was turned off in the 128-bit case, the population diversity remained considerably higher than for the 32-bit run described above, although it was less than in the 128-bit case with crossover turned on. In the 128-bit case, the population does not segregate into mutually exclusive sets — throughout a typical run, each of the terms in the fitness function provide positive values to a significant fraction of the population. Because of the relative lack of constraints in the 128-bit case, the population does not quickly become locked into a local optimum, and is able both to continue exploring longer than in the 32-bit case and to optimize each term of the fitness function more independently.

4.1.3 Is crossover effective on these functions?

As we mentioned earlier, the facts that (1) a Tanese function F evaluates points according to their parity with the j's in the terms of F, and (2) that a given F is created using partition indices of only a single order, imply that schemas of order lower than the order of F do not provide the GA with useful information. Thus crossover, one of the major strengths of the GA, is not a useful tool for recombining building blocks until schemas of order at least as high as that of the function have been discovered.

To verify this empirically, we ran the GA without crossover on 20 randomly generated 32-bit order 8 functions. The results are summarized in Table 1 under *No Xover 32*. The GA's performance was not impaired; the maximum fitness discovered and the mean population fitness are not significantly different from the runs in which crossover was used (*Original* in the table). To further verify the ineffectiveness of crossover, we ran the GA without crossover on 20 randomly generated 128-bit order 8 functions. The results are summarized under *No Xover 128* in Table 1: the performance of the GA was not significantly different from the 128-bit runs in which crossover was used (*Str Len 128*). With both the shorter and longer string-lengths (and thus with both large and small amounts of overlap), whether or not crossover is used does not seem to make any difference in the GA's performance on these functions.

[1]We counted two individuals as being different only if they were different at some significant locus — that is, at a locus in which one of the 32 js had a 1. As one would expect, with strings of length 32, every locus was significant, which is not generally the case for strings of length 128.

	Times Optimum Found	Average Max. Fit. (% opt.)	Average Gen. of Max. Fit.	Average Max. Mean Fit. (% opt.)
Original	0	88.1 (2.9)	31 (33)	85.1 (3.8)
Def-Len: 10	0	92.3 (2.9)	41 (48)	89.2 (3.0)
Str-Len: 128	19	99.97 (0.13)	150 (30)	93.6 (1.3)
No Xover 32	0	88.4 (2.7)	22 (28)	86.2 (2.6)
No Xover 128	17	99.85 (0.45)	72 (41)	93.9 (0.6)
Hill 32	0	95.4 (1.5)	-	-
Hill 128	20	100.0 (0.0)	-	-

Table 1: Summary of results of all experiments. The experiments were all performed running the traditional GA (or, in one case, hillclimbing) on randomly generated order 8 Walsh polynomials. Each result summarizes 20 runs of 200 generations each. The experiments were: (1) *Original* (replicating Tanese's experiments); (2) *Def-Len: 10* (limiting the defining length of partition indices to 10); (3) *Str-Len: 128* (increasing the string length to 128 bits); (4) *No Xover 32* (same as *Original* but with no crossover); (5) *No Xover 128* (same as *Str-Len: 128* but with no crossover); (6) *Hill 32* (hillclimbing on 32 bits); and (7) *Hill 128* (hillclimbing on 128 bits) All runs except the 128-bit runs used strings of length 32. The values given are (1) the number of times the optimum was found; (2) the maximum fitness (% of optimum) found (averaged over 20 runs); (3) the average generation at which the maximum fitness was found; and (4) the maximum population mean (% of optimum) during a run (averaged over 20 runs). The numbers in parentheses are the standard deviations.

4.1.4 Is the GA's poor performance due to deceptive functions?

Goldberg has recently shown that it is possible to construct deceptive functions by exploiting interactions among low-order terms [6], and he proposes the possibility that the Tanese functions are deceptive. We believe that the Tanese functions are not fully deceptive and that even if they are partially deceptive, deception is not the major reason that the functions are difficult. The Tanese functions are not fully deceptive because there are some schemas that do not lead the GA away from the global optimum. For example, as we discussed in Section 4.1.3, schemas of lower order than the defined order of the function provide no information to direct the GA's search.

It should be pointed out that any even-order Tanese function will have the property that $f(x) = f(x_{complement})$. For example, consider an order-2 function with one defined partition-index, $j = 0011$. $\psi_j(x)$ will be positive for the schemas ##00 and ##11, and negative for ##01 and ##10. It is possible to show that this property will hold for any even-order positive term and for any linear combination of even-order terms.

This property implies that for all the Tanese functions there will be at least two global optima that are complements of one another. Additionally, there will be at least two second most fit points, again complements of one another, and so on. This fact may have implications for whether or not the functions are deceptive. One consequence of this property is that there will be mutually exclusive families of solutions (trees of schemas) and that the GA may have trouble selecting which peak to climb (see Section 4.1.2). As we have showed, the Tanese functions are effectively higher-order than the level at which they are defined, and this fact is sufficient to account for the functions' difficulty. The fact that the functions are effectively higher-order does not imply that they are deceptive.

4.2 Why does partitioning the population seem to improve performance?

Although both the traditional and partitioned GAs performed poorly on the Tanese functions with respect to the highest fitness found, the results reported by Tanese showed that the performance of most instances of the partitioned GA was better than that of the traditional GA. This was the case even when the population was subdivided into extremely small partitions, such as 64 partitions of 4 individuals each, and, in some cases, even for 128 partitions of 2 individuals each. For functions of order higher than 4, neither the traditional nor partitioned GA's ever found the function optimum, so the difference between the two was measured only in terms of proximity to the optimum of the best fitness discovered.

As Tanese points out, these results run against conventional wisdom about GAs: it has been thought that on difficult problems a large population is needed for processing a sufficient number of schemas [4].

Tanese proposes three main reasons for this surprising result.

1. Tanese functions have a large number of local optima and the GA tends to converge on one. Each of the smaller subpopulations of the partitioned GA will converge earlier than a larger single population, but the subpopulations tend to converge on different optima, and so are able to explore a greater number.

2. After the populations converge, the major adaptive force is mutation. Mutation will be more effective in a smaller population than in a large population, since in a large population, the greater number of mutations will drive up the standard deviation of fitnesses in the population, making it less likely that fit mutations will be able to produce enough offspring to spread in the population. In the smaller population, fewer mutations will occur and the standard deviation will be lower, allowing fit mutations to spread more effectively.

3. Since smaller populations tend to be more homogeneous (for the reasons given above), fit mutations are less likely to be destroyed through crossover.

Explanation 1 seems correct, especially in light of the dynamics that were discussed in Section 4.1.2. It seems that the traditional GA quickly finds a single local optimum that satisfies a subset of partitions, and cannot go beyond that point. Since the number of different chromosomes in the population falls so quickly, the traditional GA does not use the potential for diversity that its large population offers. This explains why a number of small populations running in parallel would likely result in at least one way of satisfying the constraints of the function that was superior to the single large-population GA.

Explanation 2 relies on the assumption that large populations tend to be more diverse and have higher standard deviations than smaller ones. Since in this case, the diversity of the large population with 32-bit strings falls very quickly, this may not be a correct assumption. It seems more likely that the main factor keeping successful mutations at bay is the degree to which the problem is constrained: once a local optimum is settled upon, it is very unlikely that a single mutation (or a small set of mutations) will yield a fitter individual.

A similar point could be made with respect to Explanation 3. Given the homogeneity that results even with a large population, it seems likely that this is not a significant factor in the relative poor performance of the traditional GA.

It is very important to point out that the effects discussed here come about because of the special characteristics of the fitness functions being used — in particular, the fact that the single-order functions prevent the GA from exploiting information from lower-order building blocks through crossover. For functions in which lower-order building blocks do yield useful information, these effects might not be seen. The results and analysis for these functions cannot be assumed to be applicable to all instances of Walsh polynomials.

5 Why does hillclimbing outperform the GA on these functions?

Another surprising result reported by Tanese is that iterated hillclimbing consistently outperformed the traditional GA on the Tanese functions. On 32-bit order-4 functions, the success rate of hillclimbing was 23 out of 64 — almost eight times the success rate of the traditional GA and more than one and a half times the success rate of the best partitioned GA. Hillclimbing, like both the traditional and partitioned GAs, had a success rate of 0 on all functions of higher order, but it did consistently achieve higher fitnesses than the traditional GA. We ran stochastic iterated hillclimbing [13] on 20 randomly generated order-8 functions using the same number of function evaluations as in the GA runs. The results are summarized under *Hill 32* in Table 1: on average, hillclimbing was able to find a string whose fitness was within about 5% of the maximum, whereas the original GA was able to find a string whose fitness was only within about 12% of the maximum.

Since the nature of the Tanese functions precluded the GA from using information from lower-order building blocks via crossover, the genetic operators of crossover and mutation served mainly as a means of generating random variation — in effect, an inefficient form of hillclimbing.

6 How should the GA's performance be measured?

A major criterion by which Tanese judged GA performance was success rate, i.e., the number of runs on which the global optimum was found. This is a very strict requirement, since it does not reflect the objective difficulty (independent of any specific optimization method) of the function or how close the algorithm came to the global optimum. One large question raised by her study is, what is a reasonable way to measure the performance of GAs?

Of course, any performance measure depends to some extent on the purpose for which the algorithm is being used. However, for many problems simply recording the number of times an algorithm finds the global optimum is not adequate. The global optimum may not be known or achieving it may be an infeasible goal; in such cases, success might be measured in terms of improvement over the previously known best solution or how rapidly the algorithm can discover a solution that is "good enough" (satisficing). Under these circumstances it would be unreasonable to expect the GA to find the exact optimum or get no credit for success.

Additionally, sampling procedures like the GA are inherently nondeterministic. This implies that even if the algorithm were performing well on a particular function (and in the theoretical limit could be guaranteed to find the optimum) it might be "unlucky" in some circumstances, not reaching the exact global optimum. To some extent nondeterminism can be accounted for by performing multiple runs with different initial populations. However, it seems that for nondeterministic procedures such as the GA, it would be appropriate to use a continuous measure of success.

We believe that more quantitative measures are appropriate for comparing the performance of different optimizers on different problems, and that they should have the following properties: (1) they should be normalized so that performance can be compared on different problems and with different optimizers, (2) they should reflect problem difficulty, and (3) they should indicate how far the optimizer got, not just whether or not it reached the global optimum.

We are currently studying a metric that computes the probability that a random point would have a higher fitness than that found by the optimizer [10]. If y is the highest value found by the optimizer, f is the fitness function, and X is a random variable, then we want to find

$$y' = P(f(X) < f(y)).$$

Using this metric, the optimizer is trying to maximize y'. P can be found by computing, either analytically or empirically, the distribution of $f(X)$.

In preliminary tests using this metric, the GA and hillclimbing both do extremely well on order 8 problems, and their performance is indistinguishable from one another. This suggests that the difference between the performance of hillclimbing and the GA is insignificant when compared with the difficulty of the order 8 Tanese functions.

7 Conclusions

After examining several possible causes for the GA's poor performance on the Tanese functions, we reach several conclusions:

- Overlaps in the defined loci between different terms of the function created functions of much higher effective order than their Walsh terms suggest. This was the principle reason for the difficulty of the functions.
- The lack of information from lower-order schemas hampered crossover from contributing to the search. This was a secondary cause of the GA's poor performance, and largely accounts for the superior performance of hillclimbing.
- Long defining lengths in the non-zero partitions contributed slightly to the GAs poor performance but were not a major factor.

In addition, we feel that the success-rate performance measure is inappropriate for comparative studies of the conventional GA with variants or with other optimization methods. Under another metric, the performance of hillclimbing and GA were indistinguishable on this problem.

Tanese's results could be erroneously interpreted to imply that, in general, Walsh polynomials are difficult for GAs to optimize, and that hillclimbing will generally outperform the GA on such functions. Such a result would be a very negative one for the GA, since the Walsh functions provide a basis for any real-valued function defined on bit strings, and thus any such function can be written as a Walsh polynomial. However, the experiments and analyses reported in this paper suggest that Tanese's results should not be interpreted as a general negative verdict on the efficacy of the GA in function optimization. The functions she studied are a highly restricted subset of the class of Walsh polynomials and have several peculiar properties that make them difficult to optimize. Her results could also mistakenly be taken to imply that partitioned GAs with smaller subpopulations will always outperform traditional GAs. This may not be true for functions in which recombination of lower-order building blocks plays a major role in the search.

These results raise the important question of what functions are well-suited for the GA, and more importantly, which of these functions distinguishes the GA from other optimization methods. One clearly important factor lacking in the Tanese functions is the availability of lower-order building blocks which can combine to produce fit higher-order schemas; such hierarchical schema-fitness structure is what makes crossover an effective operator. One hypothesis is that the degree to which a function contains such structure will in part determine the probability of better optimization success for the GA than for other methods. Another hypothesized contributing factor is the degree to which the fitness landscape contains different "mountainous" regions of high-fitness points that are separated by "deserts" of low-fitness points [8]. In order to travel from a low mountainous region to a higher one, a low-fitness desert must be crossed. Point-mutation methods such as hillclimbing can have very high expected waiting times to cross such deserts (if they can cross them at all); the hypothesis is that the GA will be much more likely to quickly cross such deserts via the crossover operation.

The degree to which these factors are present in a given function may depend to a large extent on the representation chosen for the function; the role of representation in GA function optimization has been recently discussed by Liepins and Vose [11]. We are currently studying the behavior of the GA and other optimization methods on a class of functions in which the degree to which these and other factors are present can be directly varied, and we are investigating ways in which the presence of such features can be detected in a given function with a given representation. We believe that this investigation will lead to a better understanding of the classes of functions for which the GA is likely to be a successful optimization method.

Acknowledgments

This work is a natural follow-up to Reiko Tanese's thesis work. We are grateful for the interesting results that she uncovered and the clarity with which she reported them in her thesis. We regret that she was unable to collaborate with us in this work and apologize for any misinterpretations or misrepresentations. John Holland and Quentin Stout made many valuable suggestions for this research. Partial support for this project was provided by the Santa Fe Institute, Santa Fe, NM and the Center for Nonlinear Studies, Los Alamos National Laboratory, Los Alamos, NM. Support for the second author was also provided by the Michigan Society of Fellows.

References

[1] A. D. Bethke. *Genetic Algorithms as Function Optimizers*. PhD thesis, The University of Michigan, Ann Arbor, MI, 1980. Dissertation Abstracts International, 41(9), 3503B (University Microfilms No. 8106101).

[2] Stephanie Forrest. Documentation for prisoner's dilemma and norms programs that use the genetic algorithm. Technical report, University of Michigan, 1985.

[3] Stephanie Forrest and Melanie Mitchell. The performance of genetic algorithms in function optimization: Some anomalous results and their explanation. Technical report, Santa Fe Institute, Santa Fe, New Mexico, 1991.

[4] David E. Goldberg. Optimal initial population size for binary-coded genetic algorithms. Technical Report TCGA Report No. 85001, The University of Alabama, Tuscaloosa, AL 35486-2908, 1985.

[5] David E. Goldberg. Genetic algorithms and Walsh functions: Part I, a gentle introduction. Technical Report TCGA-88006, The University of Alabama, Tuscaloosa, AL, 1988.

[6] David E. Goldberg. Construction of high-order deceptive functions using low-order Walsh coefficients. Technical Report 90002, University of Illinois at Urbana-Champaign, Dept. of General Engineering, U. of Illinois, Urbana, IL, 1990.

[7] David E. Goldberg, Bradley Korb, and Kalyanmoy Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3:493–530, 1990.

[8] John H. Holland. Personal communication.

[9] John H. Holland. The dynamics of searches directed by genetic algorithms. In Y. C. Lee, editor, *Evolution, Learning, and Cognition*, pages 111–128, Teaneck, NJ, 1988. World Scientific.

[10] David Lane. Personal communication.

[11] Gunar E. Liepins and Michael D. Vose. Representational issues in genetic optimization. *Journal of Experimental and Theoretical Artificial Intelligence*, 2:101–115, 1990.

[12] Reiko Tanese. Distributed genetic algorithms. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*. Morgan Kaufmann, 1989.

[13] Reiko Tanese. *Distributed Genetic Algorithms for Function Optimization*. PhD thesis, The University of Michigan, Ann Arbor, MI, 1989.