An Empirical Evaluation of Genetic Algorithms on Noisy Objective Functions

Keith Mathias and Darrell Whitley
Department of Computer Science
Colorado State University
Fort Collins, Colorado 80523
mathiask/whitley@cs.colostate.edu

Anthony Kusuma and Christof Stork
Advance Geophysical Corporation
7409 S. Alton Ct, Suite 100
Englewood, Colorado 80209
kusuma/stork@advance.com

Abstract

Genetic algorithms have particular potential as a tool for optimization when the evaluation function is noisy. Several types of genetic algorithm are compared against a mutation driven stochastic hill-climbing algorithm on a standard set of benchmark functions which have had Gaussian noise added to them. Different criteria for judging the effectiveness of the search are also considered. The genetic algorithms used in these comparisons include an elitist simple genetic algorithm, the CHC adaptive search algorithm and the delta coding genetic algorithm. Finally several hybrid genetic algorithms are described and compared on a very large and noisy seismic data imaging problem.

Keywords: genetic algorithms, delta coding, CHC adaptive search algorithm, noisy function evaluation, seismic data imaging.

1 INTRODUCTION

Fitzpatrick and Greffenstette have provided empirical and analytic evidence suggesting that genetic algorithms exhibit a certain tolerance for noise [6]. In this paper the performance of several genetic algorithms are compared against that of a mutation driven stochastic hill-climber on a set of relatively difficult benchmark functions where Gaussian noise has been injected into the evaluation function. The genetic algorithms used in these comparisons include an elitist simple genetic algorithm, the CHC adaptive search algorithm and the delta coding genetic algorithm. These comparisons are accomplished using two different optimality metrics. In one case, we look at performance with respect to finding a noisy solution within a known range of the actual solution; in the other case, we look at each algorithm's ability to locate the true optimum. However, in both sets of comparisons the genetic algorithm only receives the noisy function evaluation value as feedback.

Additional performance comparisons involving a version of the delta coding algorithm combined with a local search algorithm were carried out for a seismic data interpretation problem known to have a noisy objective function. Because the delta coding algorithm is hybridized with a local steepest ascent algorithm, the comparisons for this application do not include the same set of algorithms used in the comparisons on the noisy benchmark functions. Instead the hybridized delta coding algorithm is compared to other hybridized algorithms that combine local and genetic search. These algorithms have been specifically designed for the seismic data interpretation problem.

Seismic reflection surveys are used to construct subsurface images of geologic beds. The images can be distorted by high levels of *ambient noise* and by effects associated with surface materials through which the seismic signal travels. However, these images can be corrected by application of "static" corrections or shifts. This problem is characterized by highly nonlinear interactions and large search spaces (e.g. 2^{6000} points is not uncommon) with many local optima [16]. The evaluation function is also noisy in the sense that the true evaluation function involves computing cross correlations between numerous seismic signals (e.g. 500) at each of 500 to 1000 time steps. To reduce the cost of evaluation, the cross correlations are sampled at a rate of 10%, thus making the evaluation function noisy and approximate.

2 BACKGROUND

In initial studies involving noisy environments [10] the delta coding genetic algorithm [18] has performed well. Delta coding begins by running a genetic algorithm while monitoring the diversity of the population. Our version of delta coding uses GENITOR [17] as the genetic search component. The GENITOR algorithm ranks the population; two offspring are then selected with a random linear bias towards the best members of the population. One offspring is produced, which replaces the worst member of the population. The new offspring is then assigned a rank in the population according to fitness. Note that the worst ranked member of the population is continually replaced; hence this position is really a kind of buffer or place holder, especially when new offspring have fitness values poorer than the members of the current population and thus are immediately replaced after the next recombination. We will refer to the *persisting population* as the normal population excluding this worst string buffer position.

One of the key components of the delta coding algorithm is a restart mechanism. Genetic search is stopped as soon as the population shows signs of losing diversity. Population diversity is monitored by testing the Hamming distance between the best and worst individuals in the *persisting population.* If it is greater than one, genetic search continues; otherwise, genetic search is temporarily suspended.

After a delta coding run is halted the search is restarted. The best solution parameters are saved as the *interim solution.* The population is then randomly re-initialized and the genetic algorithm is restarted, again monitoring the population diversity. Genetic search proceeds as normal except that the parameter substrings are decoded such that they represent a distance or *delta value* ($\pm\delta$) away from the interim solution parameters. Thus, these delta values are added to or subtracted from the interim solution parameters. The resulting parameters can then be evaluated using the original fitness function. This remaps the search space such that the interim solution is located at the origin of the hypercube. Additionally, the relationships between strings in Hamming space in the previous mapping are rearranged with respect to the schemata competitions that occur during genetic search. When the population diversity has been adequately exploited again the search is suspended and a new set of interim solution parameters are saved. This cycle is repeated until the optimum solution is located or until some specified amount of work has been expended.

Algorithm	Stochastic Hill-Climber	ESGA	GENITOR	CHC	Delta Coding
Percent Solved	20%	90%	100%	100%	100%
Average Trials	44967	28677	17367	17017	5027
Average Best	-1.73	-2.08			
Population Size	1	10	100	50	25
Crossover Rate		0.70			
Linear Selection Bias			1.25		2.0
Mutation Rate	0.02	0.005	0.01	35%	0.03

Table 1: Performance Comparisons for DeJong's F4 Noisy Test Function.

Delta coding also includes mechanisms for reducing and enlarging the size of the subpartition that is currently being searched. Each time a new interim solution is saved the number of bits used to represent each of the function parameters is altered. If the new interim solution parameters are different than those used in the previous iteration, then the parameter bit representations are decreased by one bit each. This allows the search to focus on particularly promising subpartitions of the search space. However, if the algorithm converges to the same solution (i.e. the delta values are all zero) then the parameter bit representations are increased by one. This keeps the algorithm from getting stuck in suboptimal subpartitions of the search space.

An earlier study indicated that delta coding performed better than the other algorithms considered in this study on DeJong's noisy test function (F4) when the performance of the algorithm is measured with respect to finding a noisy solution within a specific range of the known optimum. These results are reprinted in Table 1 [10]. Each algorithm was tested 30 independent times allowing a maximum of 100,000 trials for each run. The stopping criteria for these tests was to find a string with an evaluation of -2.5 returned from the noisy evaluation function (except for the ESGA tests, where a value of -2.0 was used in order to be consistent with most other previous studies). The -2.5 stopping criteria is based on the fact that the optimum is 0.0 and the added noise has a standard deviation (σ) of 1.0. No Gray coding was used to transform the binary parameter representation for the problem to avoid compromising the difficulty of the problem [11].

3 EMPIRICAL BENCHMARKS

To further evaluate the effectiveness of genetic algorithms in noisy environments we tested three genetic algorithms and a stochastic hill-climber on four benchmark test functions where Gaussian noise was artificially injected into the objective function. A binary representation is used. The functions selected for these tests included three of the larger benchmark functions used in the genetic algorithm community: the Rastrigin (F6), Schwefel (F7), and Griewank (F8) functions [13]. Gaussian noise with a mean of zero and a standard deviation of 1.0 was added to the function evaluation value for each of the functions, as in DeJong's F4 function. DeJong's F4 function was included in these tests as well. The objective of adding a Gaussian noise component to existing functions was to evaluate the performance of the algorithms for a set of functions where the performance results without the noise were already known. The equations for these test functions with the Gaussian noise component added are:

$$F4 : f(x_i \mid_{i=1,30}) = \left[\sum_{i=1}^{30} ix_i^4\right] + Gauss(0,1) \qquad x_i \in [-1.28, 1.27]$$

$$F6 : f(x_i \mid_{i=1,20}) = \left\{(200) + \left[\sum_{i=1}^{20}\left(x_i^2 - 10cos(2\pi x_i)\right)\right]\right\} + Gauss(0,1) \qquad x_i \in [-5.12, 5.11]$$

$$F7 : f(x_i \mid_{i=1,10}) = \left[\sum_{i=1}^{10} -x_i sin(\sqrt{\mid x_i \mid})\right] + Gauss(0,1) \qquad x_i \in [-512, 511]$$

$$F8 : f(x_i \mid_{i=1,10}) = \left[1 + \sum_{i=1}^{10}\frac{x_i^2}{4000} - \prod_{i=1}^{10}(cos(x_i/\sqrt{i}))\right] + Gauss(0,1) \qquad x_i \in [-512, 511]$$

3.1 Algorithm Descriptions

The mutation driven stochastic hill-climbing algorithm used in these comparisons begins by randomly generating a single binary string. Search is performed by applying bit mutation to that single string. The changes were kept only when the fitness of the resulting offspring was better than or equal to the fitness of the string before mutation. This process was repeated until the optimal solution was found or until some maximum number of trials were executed. This results in a simple stochastic hill-climbing algorithm. This algorithm was included in these tests as a benchmark for understanding the difficulty of the problems [4].

We also tested an elitist simple genetic algorithm (ESGA) [7] where the best individual in the population is always copied into the next generation. Selection in our ESGA is performed using Baker's stochastic universal sampling algorithm [1]. Two point reduced surrogate crossover was used for recombination [2] and was applied according to the probability, P_c. The probability of mutation, P_m, is the probability that a bit is flipped.

The CHC adaptive search algorithm [5] was included in these comparisons because of the algorithm's outstanding performance in other studies [5, 10]. The CHC algorithm is a generational genetic search model that employs a cross-generational selection/competition mechanism. Stings are randomly and uniformly chosen for recombination from the parent population. Offspring are held in a temporary population. Then the best N strings from the parent and offspring populations are selected for the next generation, where N is the population size ($N = 50$ for all tests performed here). If no offspring can be inserted into the population for the next generation CHC uses *cataclysmic mutation* [5] to restart the search.

Cataclysmic mutation keeps one copy of the best individual in the population. Then that string is used as a template to reinitialize the remainder of the population. These strings are formed by repeatedly mutating some percentage of bits in the template string and copying the result into the new population. A cataclysmic mutation rate of 35% was used for all of these experiments.

CHC also employs a unique crossover operator (HUX) to facilitate "heterogeneous recombination" and prevent "incest" [5]. The HUX operator compares potential mates and if the number of differing bits exceeds some threshold, uniform crossover is randomly applied to half of those positions that differ. HUX is a very disruptive crossover operator designed to scatter offspring to random points that are equidistant between parents in hyperspace.

4 Performance Comparisons Using Noisy Fitness Values to Approximate Optimality

Progress here was measured with respect to the noisy function evaluation value. Search was terminated when when the noisy function evaluation value returned to the algorithm was 2.5σ (standard deviations) below the value of the true optimal solution. Thus, if the optimal solution for a test function was 0.0 then the stopping criteria would be met when the noisy function evaluation returned a value less than or equal to -2.50. Use of the noisy function evaluation value in this manner allows a range of solutions (conditional on the noise value) to be accepted as a final solution.

Previously published algorithm comparisons have used a wide variety of metrics to track the performance of these algorithms on DeJong's F4 function. Even the uses of the noisy function evaluation criteria vary widely in research literature. Additionally, the comparisons in Section 2 do not impose the same criteria on all of the algorithms tested. For example, the progress of the ESGA was measured using a noisy function evaluation value of -2.0 while the other algorithms were compared using a value of -2.5 to indicate optimality. For the comparisons here all algorithms are measured using the same criteria (-2.5σ).

Previous genetic algorithm performance comparisons using functions included in this test suite report the use of Gray coding [5, 8, 10, 15]. However, Gray coding was not employed in these tests so that the integrity of the functions might not be compromised [11]. Previous studies also indicated that the use of Gray coding did not significantly influence the performance of an ESGA on DeJong's F4 test function [3].

For these performance comparisons the various algorithm parameters were set to match those found in previous studies [10] and then tuned by judgment so as to give the best performance possible for each algorithm. This reduces the chance of biasing the comparisons by selecting a single set of parameters that enhance the performance of one algorithm while diminishing the performance of the others. The best possible performance for these tests was defined as: 1) the most number of runs matching the *noisy* stopping criteria value for the function while performing the fewest number of recombinations; or 2) when the stopping criteria is not always satisfied, the smallest average solution in the maximum number of trials allowed for the tests.

4.1 Empirical Results and Analysis

The performance results for each of the four algorithms tested on these functions with Gaussian noise artificially injected into the evaluation function are presented in Table 2. Percent solved indicates the percentage of 30 independent runs where the noisy function evaluation value returned to the genetic algorithm reached 2.5σ below the known optimal value of the function without noise. Average trials indicates the number of recombinations performed by the algorithm for those runs that found the optimal solution. The value for noisy average best indicates the average best solution found with respect to the noisy function evaluation

Algorithm	Problem (Nb. Params)	F4 (30)	F6 (20)	F7 (10)	F8 (10)
	Total Bits	240	200	100	100
	Maximum Trials	100000	500000	200000	500000
Stochastic Hill-Climber	Percent Solved	27%	0%	0%	10%
	Average Trials	54539			311284
	σ Trials	29983			165239
	Average Best (Noisy)	-1.96	16.70	-4052.29	-1.92
	σ	0.62	4.50	111.73	0.40
	Average Best (True)	1.72	19.65	-4050.11	1.79
	σ	0.76	4.57	111.49	0.27
	Mutation Rate	0.02	0.04	0.07	0.07
ESGA	Percent Solved	40%	0%	0%	13%
	Average Trials	37486			273115
	σ Trials	28260			117362
	Average Best (Noisy)	-2.32	14.36	-4184.79	-2.24
	σ Best	0.48	3.53	21.02	0.33
	Average Best (True)	1.46	17.16	-4182.10	1.70
	σ	0.64	3.61	20.97	0.31
	Population Size	10	50	50	50
	Crossover Rate	0.70	0.80	0.60	0.90
	Mutation Rate	0.005	0.01	0.01	0.005
CHC	Percent Solved	100%	53%	100%	97%
	Average Trials	19493	321322	25243	132278
	σ Trials	8694	111752	8877	73185
	Average Best (Noisy)	-2.72	-0.21	-4192.51	-2.53
	σ		3.01		0.12
	Average Best (True)	0.62	2.01	-4189.19	1.33
	σ	0.27	1.62	0.29	1.13
	Population Size	50	50	50	50
	Cataclysmic Mutation	35%	35%	35%	35%
δ Coding	Percent Solved	100%	73%	83%	100%
	Average Trials	5989	294902	34631	80670
	σ Trials	4852	70943	6927	52035
	Average Best (Noisy)	-2.71	-2.58	-4158.9	-2.63
	σ Best		0.27	83.37	
	Average Best (True)	0.74	1.09	-4155.70	1.32
	σ	0.21	0.43	83.24	0.12
	Population Size	25	800	100	200
	Linear Selection Bias	2.00	1.90	1.90	2.00
	Mutation Rate	0.03	0.01	0.02	0.005

Table 2: Algorithm Performance Comparisons on Noisy Functions Using Noisy Evaluation Function Value Optimality Approximation Metric.

value returned to the genetic algorithm (over 30 runs) after the maximum number of trials. The noisy σ is the standard deviation associated with the noisy average best value. The true average best is the average of the *true* function evaluation value without noise for the same 30 strings evaluated for the noisy average best value.

The results presented in Table 2 indicate that the addition of Gaussian noise to these benchmark functions provides a challenging noisy test suite. The results for the ESGA on DeJong's F4 function show that the function is much harder when the noisy function evaluation value must reach -2.5σ instead of -2.0σ, as used in previous tests (See Table 1).

The mutation driven stochastic hill-climber performs worse than any of the genetic algorithms tested here on all of the functions in this test suite. This indicates that these functions are not easily hill-climbed and may also imply that the population based genetic search algorithms are able to filter noise more effectively than the stochastic hill-climber.

The results also indicate that the addition of noise and the use of the noisy function evaluation value to measure the performance of the algorithms make the Griewank (F8) function much easier to optimize than the actual function without noise. (This appears to be due to the fact that there are now multiple solutions that match the stopping criteria; it may also be significant to note that there are many near optimal solutions to the 1-dimensional version of the Griewank function that surround the global optimum.) The CHC algorithm was only able to locate the optimal solution 23% of the time for the normal F8 function without noise; CHC found a solution satisfying the stopping criteria on 97% of the runs on the noisy version of the function. Likewise, the stochastic hill-climber was not able to solve the original Griewank function without noise even a single time but was able to find a satisfactory approximation for the noisy version of the function 10% of the time. It should be noted however, that all of the average true fitnesses recorded for the F6, F7, and F8 functions are much worse than in the original versions of the problems. Thus, the actual optimization of the underlying versions of these functions without noise is much less successful when noise is injected into the function evaluation.

The noisy versions of the F6, F7, and F8 functions seem to be much harder for the delta coding algorithm than the original version of the functions. In fact, delta coding was the only algorithm to solve the original F6 and F8 functions to optimality in previous tests when Gray coding was not used and noise was not injected into the fitness functions. Delta coding was also able to consistently locate the optimal solution for the original version of the F7 function, albeit somewhat slower than the CHC algorithm. However, even when the noisy evaluation metric is used the performance of delta coding is much worse on the F6, F7 and F8 functions. Delta coding is not even able to consistently locate the optimal solution for the noisy versions of the F6 and F7 functions. Even so, delta coding still performs better than the other algorithms tested here on the F4, F6 and F8 functions with respect to the percent of the runs that were able to locate a noisy solution that satisfied the stopping criteria.

The CHC algorithm does not perform quite as well as delta coding on the noisy F4, F6, and F8 test functions. However, CHC algorithm is able to approximate optimal solutions to the noisy versions of the Rastrigin and Griewank functions with more reliability than it is able to find the true optimal solutions of the original versions of the functions. The results in a previous study with no noise [9] [10] demonstrate that the CHC algorithm was unable to locate the global optimum for the F6 and F8 functions even once in the maximum trials allotted. However, Table 2 shows that the CHC algorithm is very effective in locating noisy solutions that approximate the optimal solution for the noisy F6 and F8 functions when using the noisy function evaluation performance metric. The CHC algorithm is also the most efficient and effective algorithm for solving the Schwefel function. Additionally, the CHC algorithm proved to be very robust and required no parameter tuning.

The average true fitness values in Table 2 were obtained for these experiments to evaluate how well the solutions produced by the noisy fitness function value metric approximate the known optimum solutions for the actual functions. These values however are subject to the range of solutions that satisfy the stopping criteria; the number of solutions satisfying this criteria depends on the fitness landscape of the functions.

5 Performance Comparisons Using True Fitness Values in Noisy Optimization Environments

The following tests were accomplished using the *true* fitness value of the offspring to determine when the optimal solution was located. The only feedback provided to the algorithms however, was the noisy fitness function evaluation values.

Progress was measured with respect to each string's *true evaluation value*. As mentioned previously, using the noisy fitness function evaluation value to indicate that the algorithm has satisfied a stopping criteria allows a range of solutions to act as satisfactory solutions. This range is conditional on the noise component and the fitness landscape of the specific function. Measuring the performance of the algorithms by the *true* fitness value while providing the noisy feedback to the algorithms provides a different metric to determine how well the algorithm performs relative to the actual fitness function.

Gray coding was not used in these comparisons and the same algorithms are compared: the mutation driven stochastic hill-climber, delta coding, the CHC adaptive search algorithm and the elitist simple genetic algorithm (ESGA).

For the following performance comparisons the parameters for each algorithm were set to match those found in previous studies [10] and then tuned by judgment to give the best performance possible. The best possible performance for these tests was defined as: 1) the most number of runs locating the *true* optimal solution for the function while performing the fewest number of recombinations; or 2) when the problem is not solved consistently, the smallest average solution in the maximum number of trials allowed.

5.1 Empirical Results and Analysis

The performance results for each of the four algorithms tested on the functions in this test suite with Gaussian noise injected into the evaluation function are presented in Table 3. Percent solved indicates the percentage of the 30 independent runs that found the *true* optimal solution to the corresponding test function. Average trials indicates the number of recombinations performed by the algorithm for those runs that found the optimal solution. The value for average best indicates the average best solution found with respect to the true function evaluation (over 30 runs) after the maximum number of trials has been executed. Additionally, the lowest solution value is the lowest *true fitness value* found over the 30 runs for the corresponding test function.

Algorithm	Problem (Nb. Params)	F4 (30)	F6 (20)	F7 (10)	F8 (10)
	Total Bits	240	200	100	100
	Maximum Trials	100000	500000	200000	500000
Stochastic Hill-climber	Percent Solved	0%	0%	0%	0%
	Average Best	0.883	19.028	-4092.9	1.144
	σ Best	0.340	5.148	93.315	0.161
	Lowest Solution	0.346	10.720	-4184.5	0.686
	Mutation Rate	0.05	0.04	0.07	0.07
ESGA	Percent Solved	0%	0%	0%	0%
	Average Best	0.598	14.424	-4181.2	1.029
	σ Best	0.235	3.338	27.58	0.102
	Lowest Solution	0.308	9.070	-4189.6	0.761
	Population Size	50	50	50	50
	Crossover Rate	0.80	0.80	0.60	0.90
	Mutation Rate	0.005	0.01	0.01	0.005
CHC	Percent Solved	0%	0%	100%	0%
	Average Trials			64618	
	σ Trials			17122	
	Average Best	0.068	1.101		0.585
	σ Best	0.025	0.649		0.064
	Lowest Solution	0.023	0.218		0.426
	Population Size	50	50	50	50
	Cataclysmic Mutation		35%	35%	35%
δ Coding	Percent Solved	0%	0%	90%	0%
	Average Trials			45706	
	σ Trials			10170	
	Average Best	0.141	0.366	-4182.0	0.465
	σ Best	0.043	0.434	29.67	0.049
	Lowest Solution	0.057	0.059	-4189.8	0.391
	Population Size	25	800	100	200
	Linear Bias	2.00	1.90	1.90	2.00
	Mutation Rate	0.01	0.01	0.02	0.005

Table 3: Algorithm Performance Comparisons on Noisy Functions Using the True Fitness Function Evaluation Value for Optimality. The *true* optimal solution for the F4, F6 and F8 functions is 0.0. For the F7 function the *true* optimal solution is -4189.82764 (conditional upon the machine architecture).

The results presented in Table 3 indicate that the addition of Gaussian noise to these benchmark functions provides a very difficult test suite when the stopping criteria for the search is locating the *true* global optimum.

The stochastic hill-climber performs worse than all of the genetic algorithms tested here. The F4 results in Table 3 also indicate that the performance metric based on the *true* function evaluation value is much more difficult to satisfy than the stopping criteria using the noisy function evaluation value. In fact, the results in Table 1 would seem to indicate that the genetic algorithms tested here are consistently able to solve the F4 function. However, the results in Table 3 indicate that none of the algorithms tested here are able to locate the true optimal solution for the F4 function within the maximum trials allotted. This seems to imply that while the genetic algorithms are more tolerant of noise than the stochastic hill-climber tested here they are not able to filter out all of the noise and consistently locate the optimal solution. This is consistent with the observation that the noise component of the evaluation function may dominate the feedback that the genetic algorithm receives as the population converges on strings with evaluations that are similar to the string representing the optimal solution.

These results also indicate that the CHC and delta coding genetic algorithms perform significantly better than the other algorithms tested here. CHC and delta coding are the only two algorithms able to solve the F7 function and both algorithms reach much better average solutions on the other functions of the test suite. This may suggest that these iterative algorithms (i.e. delta coding and CHC) are very effective in quickly locating solutions that are in the vicinity of the optimal solution. Nevertheless, their ability to locate the true optimal solution is diminished as the variance in string evaluations becomes dominated by the noise component of the evaluation function. Neither of these two algorithms is clearly best for locating the *true* optimal solution in noisy optimization environments.

In previous comparisons where noise was not interjected, delta coding was the only algorithm tested that was able to consistently locate the optimal solution to the Rastrigin and Griewank functions without the use of Gray coding [10]. Additionally, delta coding performed significantly better than the other algorithms tested on the F4 function using the noisy function evaluation value metric. However, in these comparisons the delta coding algorithm was not able to locate the *true* optimal solution in any of the F4, F6, or F8 experiments in the trials allotted.

6 Discussion of Empirical Tests

The results presented here indicate that this test suite is resistant to hill-climbing. Noisy function evaluation significantly impacts the performance of the stochastic hill-climber tested here while the noisy feedback has less severe effects on the performance of the population based genetic search algorithms. During the early stages of genetic search noise seems to be effectively filtered so that the genetic algorithm can adequately sample the fitness landscape. However, as solutions that are competitive with the global optimum are discovered by the search, the noise component of the fitness function becomes an increasingly significant part of the individual's fitness. As the noise component begins to dominate the feedback to the algorithm it becomes more difficult to approximate the underlying fitness landscape. At this point in the search process strategies that more randomly sample the immediate neighborhood of solutions in the current population may have an advantage in locating the global optimum. This may explain why CHC performed so well in the experiments that used the true fitness value as the performance metric. One enhancement that might improve the performance of these algorithms with respect to locating the global optimum is to re-evaluate strings multiple times as the noisy fitness function evaluation values become comparable to the global optimum. This would help filter the noise and possibly allow the algorithms to better approximate the true fitness of the strings.

Of the algorithms tested here delta coding and the CHC adaptive search algorithm appear to be more effective in quickly locating quality solutions for the corresponding functions regardless of the performance criteria. Delta coding appears to perform slightly better than CHC when the noisy function evaluation value is used as a stopping criteria. However, the CHC adaptive search algorithm performed slightly better when the *true* fitness value was used to measure the performance of the algorithms.

The two sets of criteria used in these experiments as performance metrics demonstrate the dilemma of designing an adequate method for evaluating an algorithm's ability to search a noisy function. Requiring that the algorithm locate solutions that are within some σ limits of the optimal solution is unsatisfactory as the number of solutions that meet this criteria is dependent on the fitness landscape of the underlying function.

Finally, it is important to note that these comparisons evaluate the performance of these algorithms for a small number of functions which have had only one type of noise artificially injected into the fitness evaluation function. Thus, any conclusions are speculative; this motivates the need to develop a larger test suite of functions that include the injection of different types of noise.

7 An Application: Geophysical Static Corrections

Images of subsurface strata obtained from seismic reflection surveys are often distorted due to *ambient noise* and the fact that different seismic signals travel through dissimilar materials near the earth's surface. The geophysical static corrections problem is defined as the search for the temporal offsets necessary to correct the signals obtained in the reflection surveys resulting in an accurate geologic strata image. This problem is very difficult because of the inherent signal noise, very large search spaces with many local optima and highly non-linear characteristics of the search space. Conventional methods for finding static corrections often result in unacceptable solutions when the level of ambient noise present is high or when severe surface heterogeneities exist. Therefore, global search methods that are not sensitive to *noise* have been explored.

Attempts to use genetic algorithms for solving the static corrections problem have previously proven to be computationally ineffectual [14, 19]. Even attempts to combine genetic search with local search have proven to be computationally expensive [16]. Nonetheless, these hybrid genetic algorithms have resulted in significantly improved solutions. These hybrid genetic algorithms use a *noisy* evaluation function that computes only a small percentage (i.e. 10%) of the cross-correlations between the signals in the reflection survey. This also introduces additional noise in the evaluation function.

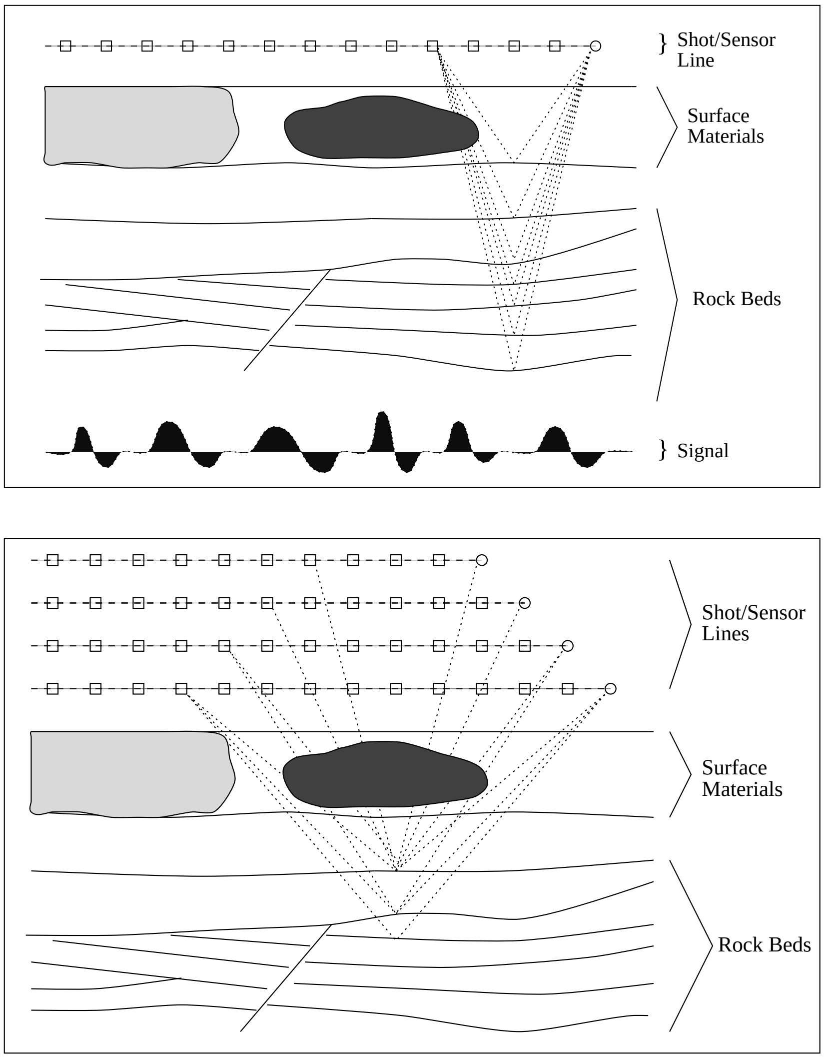

Figure 1: Sensor Shot Signals (1a: top) and Common Mid Point Gather (1b: bottom).

7.1 Problem Description

Seismic reflection surveys consist of a series of signals generated by a detonation (shot) and recorded at a series of sensor sites. Figure 1a illustrates the reflection signals that a single sensor (□) would collect for a single shot (∘) as well as the waveform corresponding to the signal. Each of the sensors along the sensor-line would collect a similar set of reflection signals for this shot. A number of shots would be detonated over time, moving the shot site to the adjacent sensor site for each new signal collection (top of Figure 1b).

The signals collected at each sensor can be filtered and re-combined so that all reflections occurring at common midpoints are grouped into corresponding synthesized traces. Figure 1b illustrates the reflection signals that would be grouped into a common midpoint (CMP) gather. These synthesized traces are then rotated 90° and grouped together to form a distorted image of the subsurface strata called a *stack*. Applying the true temporal corrections to each signal results in an accurate image of the subsurface strata.

The traces in a CMP gather are affected by two types of temporal phenomena. The first temporal effect is due to the difference in distance that the signals travel from various sensor/shot pairs. The temporal corrections for this phenomenon are collectively known as the *normal move out* (NMO) and can be approximated since the temporal differences between signals at a CMP are directly related to the distance traveled. The second type of temporal phenomenon is a result of the difference in materials at the earth's surface. Dense materials tend to increase the velocity of the signal while porous and loosely packed materials tend to delay the signal. Since the materials beneath the detonation and sensor sites may vary, two independent "static" corrections are applied to each signal. The static corrections (one for each signal) make up the parameters that we wish to optimize.

7.2 Algorithm Descriptions

Our initial experiments, as well as the experiences of other researchers, indicated that some combination of steepest ascent and genetic search is needed to achieve adequate performance on this problem. We therefore compared the performance of four types of hybrid genetic algorithms on the geophysical static corrections problem. The descriptions of those algorithms follow.

7.2.1 The GENITOR Hybrid Algorithm

The GENITOR hybrid algorithm tested here uses a population of binary strings. When decoded, these strings represent the static corrections to the signals in the seismic reflection survey. The strings are initially seeded in the population using a small set of biased correction models (e.g. 5) that are calculated from *a-priori* knowledge about the signal correlations. These models are then improved by iterative application of waveform steepest ascent. The improved models are then converted to binary representations to serve as templates for initializing the population. The binary strings in the population are then formed by repeatedly mutating 15% of the bits from the template solutions and placing the results in the population (i.e. cataclysmic mutation). None of the template strings are included in the population initially.

A single NMO is used for the evaluation of all offspring. This NMO is computed by averaging the NMOs of the template solutions. The evaluation function returns the cross-correlation power of the stack. Genetic search proceeds without application of any local search methods until the population converges. Genetic search is executed using the GENITOR algorithm. When the population has converged, waveform steepest ascent is iteratively applied to the best solution in the population to improve the quality of the final solution.

7.2.2 The Staged Hybrid ESGA Model

The design of the *staged* hybrid ESGA resembles that of the hybrid GENITOR algorithm in that the two algorithms both intelligently seed their initial populations from solutions that have been improved by iterative application of waveform steepest ascent. They both also improve their final solutions by iterative application of the steepest ascent algorithm. However, the staged hybrid ESGA employs a floating point representation for the population individuals. This same representation was used by Kusuma and Stork [16], where each floating point number corresponds to a single static correction. Additionally, *all* of the individuals in the initial population were developed directly from *a-priori* signal correlation knowledge and waveform steepest ascent was iteratively applied to all of the members of the population. This allows a unique NMO to be paired with each individual.

After the initial population has been formed, genetic search is allowed to continue **uninterrupted** for 10 generations. After every 10^{th} generation one iteration of waveform steepest ascent is applied to all individuals in the population. This pattern is repeated until the 40^{th} generation when the waveform steepest ascent algorithm is applied iteratively to the entire population again.

7.2.3 The Staged Hybrid GENITOR Algorithm

The *staged* hybrid GENITOR algorithm employs binary string representations as used in the GENITOR hybrid algorithm. The initial population and initial NMO model are developed using the same process used in the GENITOR hybrid algorithm. Then genetic search is allowed to proceed. However, after some number of predefined trials have been executed (e.g. 10,000) the genetic search is temporarily halted.

The best solution at the end of each stage of the genetic search is converted to a floating point representation and inserted into the original set of models used to seed the population, displacing the worst model. At this point, one iteration of waveform steepest ascent is applied to the floating point models. These models are then used as templates to re-seed the population for the next stage of genetic search in the same manner as they were used in the initial seeding process. A single NMO is also developed for the evaluation of the offspring using the same averaging technique used initially. Genetic search is then restarted.

This cycle of genetic search followed by local search is repeated until some number of total trials have been executed. When the final stage of genetic search is completed, the best solution in the population is inserted into the floating point model set and waveform steepest ascent is iteratively applied to improve the final solution.

7.2.4 The Hybrid Delta Coding Algorithm

The hybrid version of delta coding (Figure 2) that we tested on the static corrections problem employed binary strings. The strings were initialized using the same procedures described for the GENITOR hybrid and staged hybrid GENITOR algorithms. The initial NMO used for evaluating offspring was also developed using the same averaging technique as the other two algorithms. After the population is seeded, genetic search using the GENITOR algorithm is allowed to proceed for a predetermined number of trials. The normal method of testing population diversity using the Hamming distance metric was not used for testing on this problem. At the end of the genetic search iteration the best solution in the population is converted to a floating point representation and a single iteration of waveform steepest ascent is applied. This solution, along with the corresponding NMO is saved as the interim solution. Each string in the following iteration is decoded and referenced with respect to this interim solution.

Population re-initialization is performed by producing binary strings where some percentage (i.e. 15%) of the bit positions in each string are initialized to ones. These positions are chosen randomly. One string consisting of all zeroes is seeded into the population as well. This string represents the interim solution in the delta coded format. The other strings in the population, when decoded, represent static offsets close to the interim solution

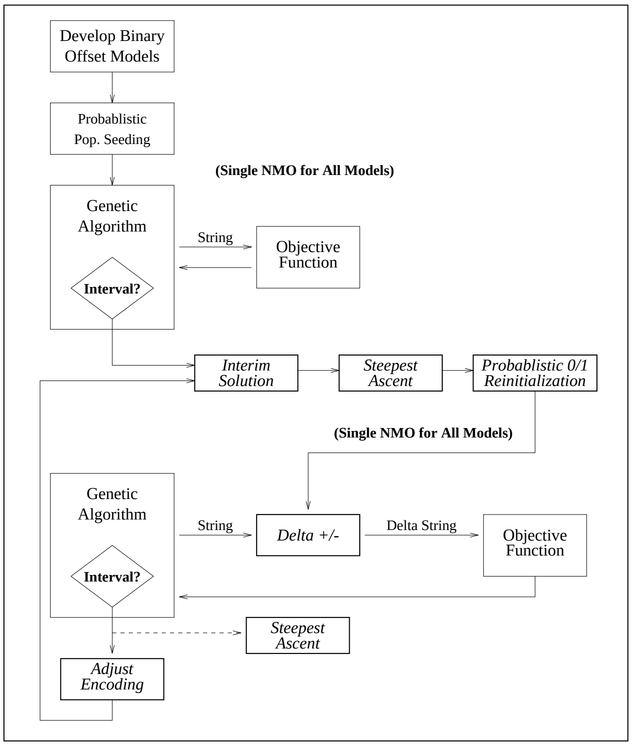

Figure 2: Hybrid Delta Coding Algorithm.

in numeric space.

After the population is re-initialized genetic search resumes. All offspring parameters are decoded as delta values ($\pm\delta$) which are added to the corresponding parameter in the interim solution. This allows the algorithm to explore a new mapping of the search space while employing the same objective function throughout the search. This iteration is allowed to proceed for the same number of predefined trials as the first stage of genetic search.

When the delta iteration is completed, the best solution is again converted to floating point format and waveform steepest ascent is applied. This results in a new interim solution. The population is then re-seeded using the method described for the second iteration of search except that the parameter representations are altered. The number of bits used to represent each parameter is reduced by one. This decreases the range of the δ values by half. When the population has been re-initialized, genetic search is resumed. This cycle is repeated until some number of trials have been executed. The final static corrections model is determined by iterative application of waveform steepest ascent to the best solution in the final population.

The hybrid delta coding algorithm provides the same separation of effort between the local and genetic search components as the other staged algorithms. It also saves computational effort by applying the waveform steepest ascent algorithm to a limited number of models. The delta coding algorithm also provides some additional advantages including a reduced population size (representing an additional time savings), reduced search space, and the exploration of multiple mappings of the search space over the course of the search.

7.3 Empirical Results and Analysis

Figure 3 shows typical runs of the basic GENITOR hybrid algorithm, a staged hybrid ESGA, a staged GENITOR hybrid algorithm and a staged delta coding algorithm for the statics corrections problem. The evaluation value along the Y axis represents the power of the stack (which is being optimized) and the X axis indicates the time in CPU seconds as executed on a SPARC II workstation. The strategies shown here have been used for comparison due to their performance in other studies [12].

The staged hybrid GENITOR algorithm produces the best overall results for the static corrections problem. The objective function values produced by the staged hybrid GENITOR algorithm are approximately 7% better than those yielded by the staged hybrid ESGA and are produced approximately three hours faster. However, the delta coding algorithm produces results that are approximately 5% better than the staged hybrid ESGA but does so almost a full five hours faster.

Analysis of the performance curves that are associated with the staged search algorithms reveals that the sharp inclines indicating rapid improvement in the solutions being found are associated with the waveform steepest ascent stages of the search. The intervals indicating slow improvement are associated with the genetic stages of the search. This might lead one to believe that skipping the genetic search stages and iteratively executing local search would be advantageous. However, the genetic search stages actually support the local search by generating new points in different subpartitions of the search space, thus providing the local ascent algorithm with new points from which to climb. Furthermore, the best model associated with the waveform steepest ascent search may be a local optimum and if so, this point would not be improved by additional local search.

8 CONCLUSIONS

The results presented here indicate that noisy function evaluation has more impact on the stochastic hill-climbing algorithm than on population based genetic algorithms. Furthermore, dealing with noise during the early stages of genetic search seems to be relatively easy. But as the various genetic algorithms begin to generate solutions that were comparable to the global optimum, the noise component of the objective function becomes an increasingly significant factor during evaluation. At this point in the search process strategies that more randomly sample the immediate neighborhood of solutions in the current population may have an advantage in finding the global optimum. This may explain why CHC performed so well in our experiments.

We are thus left with a dilemma; the criteria we used may not be a good way to evaluate an algorithm's suitability for searching noisy functions; on the other hand, just requiring that an algorithm find some solution within 2 or 2.5 standard deviations of the evaluation of the global optimum also seems unsatisfactory. One alternative would be to use multiple evaluations (or more costly, but more actuate evaluation) as the search narrows its focus. This would provide a more direct means of canceling the noise at a point in the search when the variance over the set of solutions in the population may be dominated by the noise.

The delta coding algorithm proved to be slightly inferior to the staged hybrid GENITOR algorithm for the seismic "statics" problem in terms of the final results obtained, but delta coding found good solutions faster than the other algorithms we tested. We believe that

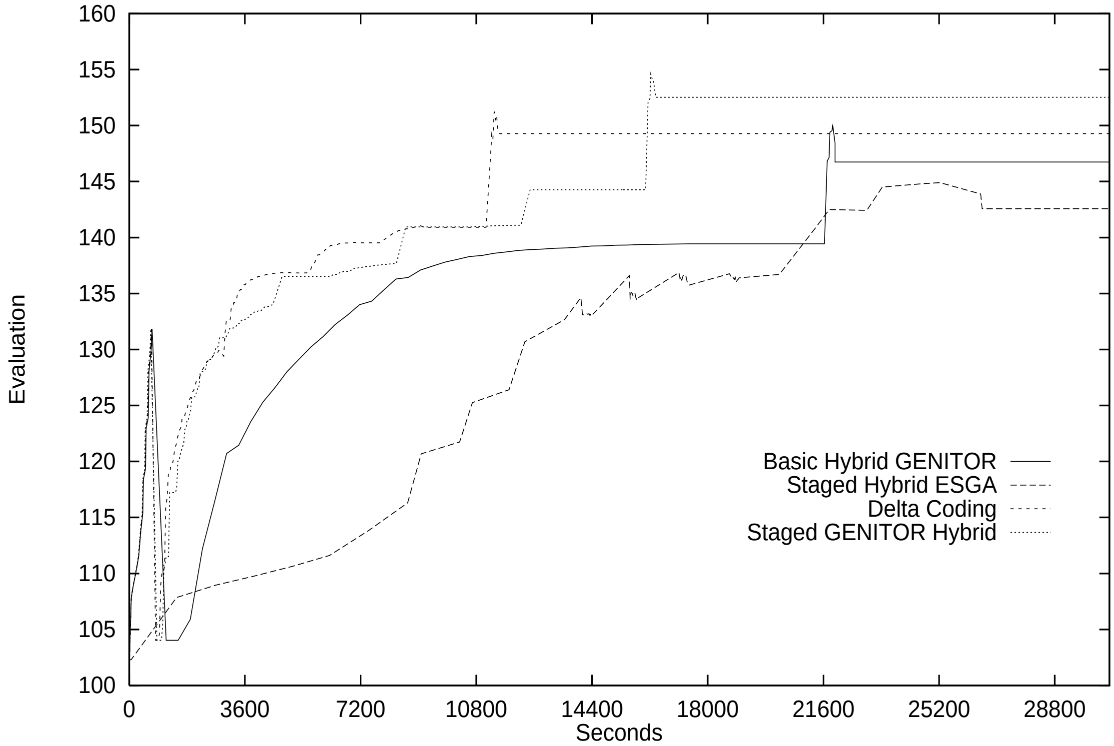

Figure 3: Timing Performance Comparisons.

delta coding was at a disadvantage in these experiments because it works with a single NMO rather than an average of several NMOs. Nevertheless, being able to generate good solutions quickly can also be advantageous in many application domains.

9 ACKNOWLEDGMENTS

This research was supported in part by a grant from the Colorado Advanced Software Institute and from NSF Grant IRI-9312748.

References

[1] James Baker. Reducing Bias and Inefficiency in the Selection Algorithm. In John Grefenstette, editor, *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference*, pages 14–21. L. Erlbaum Assoc., 1987.

[2] Lashon Booker. Improving Search in Genetic Algorithms. In Lawrence Davis, editor, *Genetic Algorithms and Simulated Annealing*, chapter 5, pages 61–73. Morgan Kaufmann, 1987.

[3] R. Caruana and J. Schaffer. Representation and Hidden Bias: Gray vs. Binary Coding for Genetic Algorithms. In *Proceedings of the Fifth International Conference on Machine Learning*. Morgan Kaufmann, 1988.

[4] Lawrence Davis. Bit-Climbing, Representational Bias, and Test Suite Design. In L. Booker and R. Belew, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 18–23. Morgan Kauffman, 1991.

[5] Larry Eshelman. The CHC Adaptive Search Algorithm. How to Have Safe Search When Engaging in Nontraditional Genetic Recombination. In G. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 265–283. Morgan Kaufmann, 1991.

[6] J. Michael Fitzpatrick and John Grefenstette. Genetic Algorithm in Noisy Environments. *Machine Learning*, 3:101–120, 1988.

[7] David Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[8] V. Gordon and D. Whitley. Serial and Parallel Genetic Algorithms as Function Optimizers. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 177–183. Morgan Kauffman, 1993.

[9] Keith E. Mathias and L. Darrell Whitley. The Delta Coding Algorithm: Comparisons to CHC, SGA, GENITOR and Hill Climbing. Submitted to the Journal of Evolutionary Computation for publication.

[10] Keith E. Mathias and L. Darrell Whitley. Initial Performance Comparisons for the Delta Coding Algorithm. In J. D. Schaffer, editor, *Proceedings of the IEEE International Conference on Evolutionary Computation*, 1994.

[11] Keith E. Mathias and L. Darrell Whitley. Transforming the Search Space with Gray Coding. In J. D. Schaffer, editor, *Proceedings of the IEEE International Conference on Evolutionary Computation*, 1994.

[12] Keith E. Mathias, L. Darrell Whitley, Christof Stork, and Tony Kusuma. Staged Hybrid Genetic Search for Seismic Data Imaging. In J. D. Schaffer, editor, *Proceedings of the IEEE International Conference on Evolutionary Computation*, 1994.

[13] H. Muhlenbein, M. Schomisch, and J. Born. The Parallel Genetic Algorithm as Function Optimizer. In L. Booker and R. Belew, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 271–278. Morgan Kauffman, 1991.

[14] J. Scales, M. Smith, and T. Fischer. Global Optimization Methods of Highly Nonlinear Inverse Problems. In *Proceedings of the International Conference on Numerical Aspects of Wave Properties and Phenomenon*, 1990. France.

[15] J. David Schaffer, Richard A. Caruana, Larry J. Eshelman, and Rajarshi Das. A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 51–60. Morgan Kauffman, 1989.

[16] C. Stork and T. Kusuma. Hybrid Genetic Autostatics: New Approach for Large-Amplitude Statics with Noisy Data. In *Proceedings of SEG* 62^{nd} *Annual International Meeting*, pages 1127–1131. Society of Exploration Geophysicists, 1992.

[17] L. Darrell Whitley. The GENITOR Algorithm and Selective Pressure: Why Rank Based Allocation of Reproductive Trials is Best. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 116–121. Morgan Kauffman, 1989.

[18] L. Darrell Whitley, Keith Mathias, and Patrick Fitzhorn. Delta Coding: An Iterative Strategy for Genetic Algorithms. In L. Booker and R. Belew, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 77–84. Morgan Kauffman, 1991.

[19] W. Wilson and K. Vasudevan. Application of the Genetic Algorithm to Residual Statics Estimation. *Geophysical Res. Let.*, 18(12):2181–2184, 1991.