Department of Computer Science

Remapping Subpartitions of Hyperspace Using Iterative Genetic Search

Keith Mathias and Darrell Whitley

Technical Report CS-94-101

January 7, 1994

Colorado State University

Remapping Subpartitions of Hyperspace Using Iterative Genetic Search

Keith Mathias and Darrell Whitley

Department of Computer Science
Colorado State University
Fort Collins, Colorado 80523
mathiask, whitley@cs.colostate.edu

Abstract

Various strategies for remapping hyperspace associated with two *iterative* genetic algorithms are reviewed, compared and empirically tested on a particular "real-world" problem. Both algorithms reduce the hypercube at each iteration such that each new mapping samples only subpartitions of the search space. *Delta coding* remaps the search space at each iteration by sampling the subspace numerically adjacent to the previous solution. *Canonical delta folding* canonically reorders and folds hyperspace producing new mappings that sample subpartitions distributed across the search space. Empirical results suggest that more local sampling strategies generally produce better solutions and that identification of "good" mappings is difficult.

Remapping Subpartitions of Hyperspace Using Iterative Genetic Search

Abstract

Various strategies for remapping hyperspace associated with two *iterative* genetic algorithms are reviewed, compared and empirically tested on a particular "real-world" problem. Both algorithms reduce the hypercube at each iteration such that each new mapping samples only subpartitions of the search space. *Delta coding* remaps the search space at each iteration by sampling the subspace numerically adjacent to the previous solution. *Canonical delta folding* canonically reorders and folds hyperspace producing new mappings that sample subpartitions distributed across the search space. Empirical results suggest that more local sampling strategies generally produce better solutions and that identification of "good" mappings is difficult.

1 Introduction

The goal of genetic search is to exploit statistical information about hyperplane partitions by using the relative fitness of strings sampled from those partitions to allocate reproductive opportunities. The resulting selective pressure should direct the search toward partitions of hyperspace representing above average solutions. However, hyperplane feedback can mislead the search so that it converges to a sub-optimal solution. Liepins and Vose (1990) argue that for any specific problem there exists some transformation of the space such that all hyperplane competitions lead to the global optimum (c.f., Battle and Vose, 1991). Unfortunately, there is currently no general method for transforming an arbitrary function so as to make it easier to optimize. Furthermore the space of all possible mappings is much larger than the original search space.

However, the search for finding "easier" mappings of hyperspace need not yield the optimal mapping. Any number of remappings may potentially decrease the difficulty of a search space as compared to a single mapping, as is normally used in genetic search. The use of an iterative genetic algorithm provides an ideal opportunity for exploring multiple mappings of hyperspace, since a new mapping can be implemented with each new restart. However, a method for identifying "good" mappings is necessary to exploit these transformations.

Various strategies for remapping hyperspace associated with two *iterative* genetic algorithms are reviewed, compared and empirically tested on a particular "real-world" problem. Both algorithms reduce the hypercube from one iteration to the next. As a result, each new mapping only samples subpartitions of the entire search space. *Delta coding* remaps the search space at each iteration by sampling the subspace numerically adjacent to the previous solution. *Canonical delta folding* canonically reorders and folds hyperspace; the result is a mapping composed of points selected from distributed subpartitions of the search space.

Empirical results suggest that local sampling strategies are generally more effective in finding improved problem solutions than more distributed sampling strategies. Empirical results

also indicate that it is difficult to evaluate how "good" a particular mapping of hyperspace is based solely on performance. This is because the variance in fitness associated with different transformations of the space cannot be distinguished from the variance observed across different population samples using a single mapping.

2 The Delta Coding Algorithm

Delta coding (Whitley, Mathias and Fitzhorn 1991) uses GENITOR (Whitley, 1989) as the basic engine for genetic search. The genetic algorithm is executed in a normal fashion on the first iteration except that the population diversity is monitored by measuring the Hamming distance between the best and worst members of the population. If the Hamming distance is less than 2, the search is temporarily stopped and the best solution is saved as the *interim solution*. A new population is created randomly and GENITOR is restarted. Each parameter is subsequently decoded, however, as a *delta value* ($\pm\delta$) that is added (or subtracted) to the interim solution.

At the termination of each delta iteration the *delta coding* algorithm uses information about the interim solution to trigger operations that sometimes reduce or increase the number of bits per parameter in the string. This permits the search to *temporarily* focus on a reduced area of the search space. The method for delta parameter decoding remains the same for all iterations. If the sign bit of the delta value is zero, directly add the delta value to the interim solution parameter. If the sign bit of the delta value is one, complement all other bits and add the result to the interim solution parameter. All addition is performed mod 2^b, where b is the number of bits in the original parameter encoding.

Each iteration effectively remaps the search space by constructing a new hypercube with the interim solution at the origin. This has two effects: 1) each new hypercube creates a new set of hyperplane competitions (via remapping) and 2) a different set of points may be sampled if the interim solution changes or if the parameter representation is changed (Figure 1).

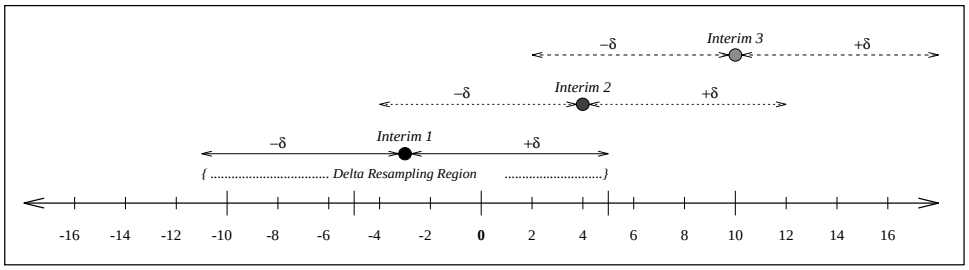


Figure 1: Points Sampled In Numeric Space Using Delta Coding.

3 The Canonical Delta Folding Algorithm

Canonical delta folding uses the same iterative mechanisms present in the original delta coding algorithm but employs a more general method for remapping hyperspace. This remapping is applied in two phases. First, the space is placed in canonical order by *exclusive or-ing* (denoted

$\oplus$) each string in the space with the interim solution. This preserves the same hyperplane partitions, but "rotates" hyperspace to place the interim solution at the origin. Second, the search space is folded. The folding remaps the strings in the new search space, changing the relationships between strings and hyperplane partitions in the space. This transformation is given by $\delta_S = [C_{f1}(S \oplus I)]$ where S is the string from the population being remapped, δ_S is its *canonical delta fold* representation, I is the interim solution bit string, and C_{f1} is the bitwise complement function applied conditional upon the "fold bit" being one. Subscript "f1" denotes the fold bit as bit 1. The invertible point-to-point mapping, transforming strings in the delta fold hypercube back to the original parameter space, is represented as $S = [(C_{f1}(\delta_S)) \oplus I]$.

The folding mechanism applied to a 3 bit problem in canonical delta folding via C_{f1} can be represented in matrix form where a string S, in the original coding of the space is related to δ_S, a string in the canonical delta fold space in relation to the interim solution string, I, by:

$$(I \oplus S) \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \delta_S$$

All vectors are row vectors and addition is mod 2 (equivalent to $\oplus$). However, as any bit may serve as the fold bit, the folding function can be generalized to C_{fx}, where x represents the position of the fold bit. Changing the fold bit merely means that the space is folded along another dimension. Therefore, the generalized matrix for mapping strings from the original coding of the space to the canonical delta version of the search space has 1 bits in the row corresponding to the fold bit, 1 bits on the diagonal and 0 bits everywhere else. A more detailed description of canonical delta folding is given by Mathias and Whitley (1993).

3.1 Focusing the Search: Shrinking the Hypercube

Canonical delta folding, like delta coding, focuses the search at each iteration by reducing or expanding the number of bits used to encode parameters. While this mechanism remaps the search space in both strategies to some degree, it is designed to remap hyperspace in a predetermined fashion in delta folding. When the hypercube is reduced, different subpartitions of hyperspace may be sampled depending on how hyperspace is remapped. The following sampling and remapping strategies are examples of the transformations used by delta folding.

Figures 2 through 4 represent the sampling distribution of hyperspace for each of the three strategies. The lines, from top to bottom, represent a reduction of the search space by 1 bit. The raised portion of each line represents the subpartitions of the space currently being sampled. Strategy 1 samples information local to the interim solution, located at 0, and wraps around the original space to sample information about it's global complement. The algorithm for mapping strings in the reduced space back to the original space is given by:

IF (bit1 of $\delta_S == 1$) $\Longrightarrow$ $(C_{f1}(\delta_S))$ and pad to the left with 1's
ELSE $\Longrightarrow$ Pad δ_S to the left with 0's
XOR Padded-string with current interim solution

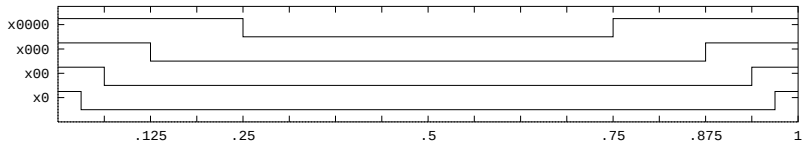


Figure 2: Points Sampled in Hamming Space for Remapping Strategy 1.

The second strategy is designed to sample the global complement of the interim solution in the original problem space and the local complement in the reduced space. It still folds the global complement of the interim solution to a position adjacent to the interim solution in Hamming space. Figure 3 shows the subpartition sampling pattern for this strategy and the algorithm is outlined below. References to "bit1" and "bit2" refer to the first and second bit of δ_S, respectively.

IF (bit1 of δ_S == 0) $\Longrightarrow$ Pad δ_S to the left with 0's
ELSE IF (bit1 == 1) && (bit2 == 0) $\Longrightarrow$ $(C_{f1}(\delta_S))$ and Pad to the left with 1's
ELSE IF (bit1 == 1) && (bit2 == 1) $\Longrightarrow$ Pad δ_S to the left with 0's
XOR Padded string with current interim solution

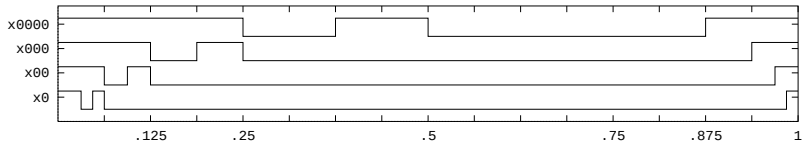


Figure 3: Points Sampled in Hamming Space for Remapping Strategy 2.

The third strategy is similar to strategy 2 except it does not always resample the global complement of the interim solution, but instead samples the current complement of the reduced form of the interim solution and the complement of the interim solution at the level above (i.e. with one additional bit added to the representation). Therefore, in the first sampling the global and local complements are sampled. At the next level of reduction the local complement at the current level and the local complement of the current interim solution with one additional bit are sampled. The points sampled in canonic space over multiple reductions of the search space for this strategy are shown in Figure 4.

IF (bit1 == 1) && (bit2 == 1) $\Longrightarrow$ Prefix with 1
IF (bit1 == 1) && (bit2 == 0) $\Longrightarrow$ $(C_{f1}(\delta_S))$ and Prefix with 0
Pad prefixed-string to the left with 0's
XOR Padded string with current interim solution

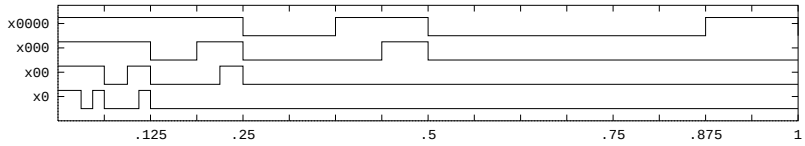


Figure 4: Points Sampled in Hamming Space for Remapping Strategy 3.

Strategies 2 and 3 sample more information that is local to the interim solution than strategy 1. Strategy 1 restricts it's sampling sets to the information in Hamming space that is directly adjacent to the interim solution and it's complement.

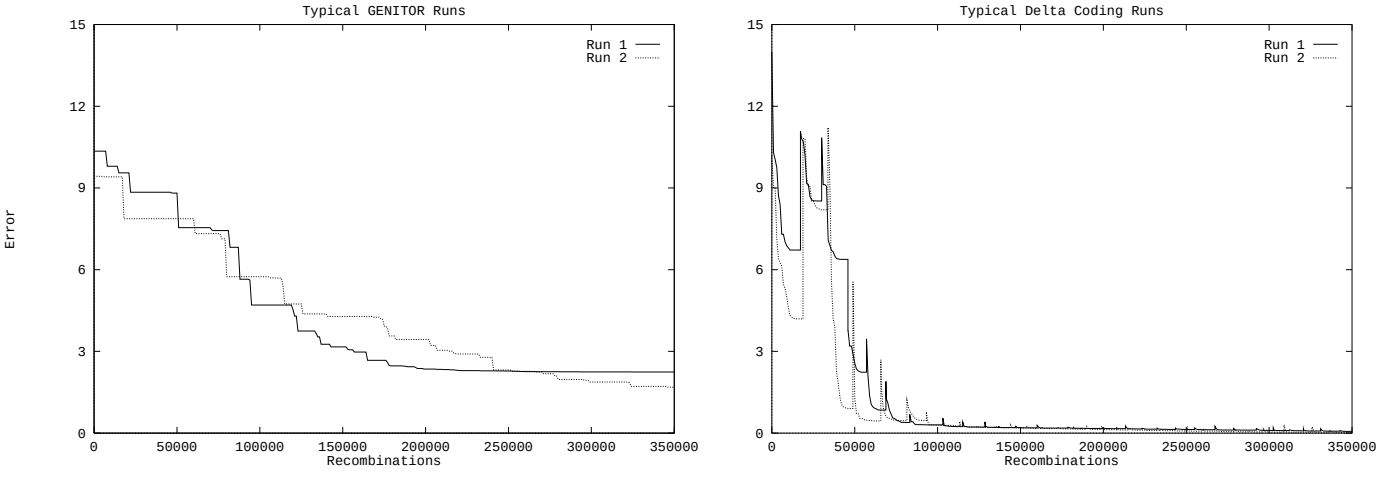
4 Empirical Performance

In this paper, delta coding and canonical delta folding are empirically tested and compared using a rigid body transformation problem. The goal of the search is to find optimal rigid body transformations to map a set of points in three dimensions (3-D) onto a target set of points in a 2-D domain. Eight transformation parameters were encoded using 12 to 14 bits each, for a total of 104 bits representing: translation along and rotation around three axes, a uniform scale, and perspective. A fitness function designed to measure Euclidean distance between the target and transformed points was used to evaluate the parameter sets. Solutions with zero error are known to exist.

4.1 Remapping and Subpartition Sampling Strategies

As shown previously (Whitley, et. al., 1991), delta coding finds more precise solutions in fewer trials using smaller populations than the GENITOR and the parallel GENITOR II algorithms. Representative runs have been chosen for Figures 5 through 7 to understand and compare the behaviors of the algorithms. These plots track the best individual in the population over a single search. As representative samples they reflect the general behavior and performance exhibited by each strategy and the general variance between the runs.

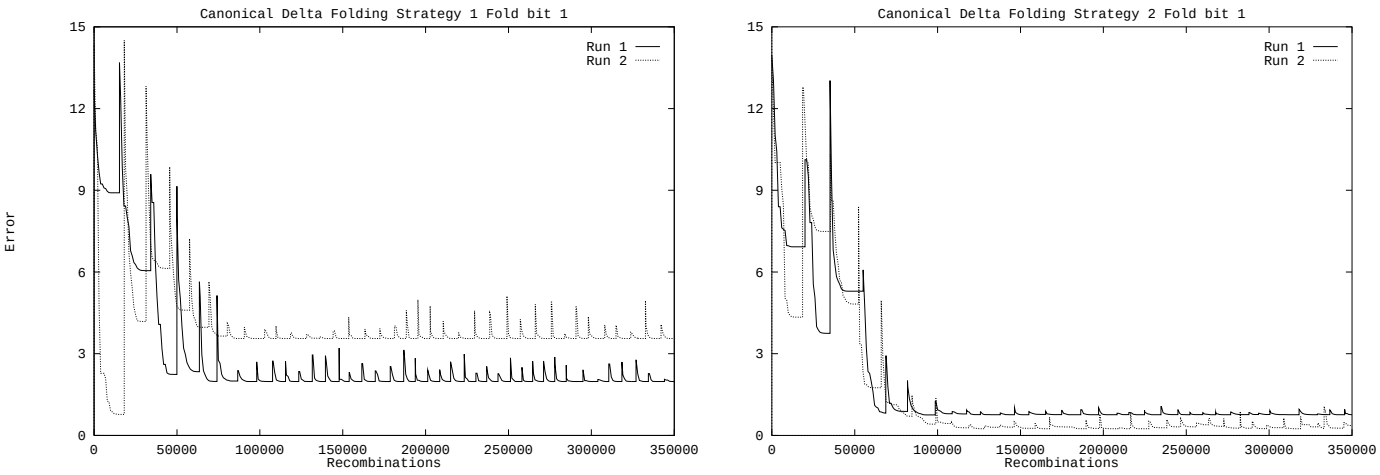
The GENITOR runs in Figure 5a were tuned for best performance, using a population of 5000 and a selection bias of 1.25. The delta algorithms exhibited in Figures 5b through 9a used a population of 500 and a bias of 2.0. All delta coding and canonical delta folding searches used a lower limit of 6 bits/parameter, unless otherwise specified.



(a) (b)

Figure 5: Typical **GENITOR** and **Delta Coding** Searches for 3-D Transformations.

The typical performance and behaviors for the canonical delta folding algorithm using the strategies presented in section 3.1 and C_{f1} are shown in Figures 6a, 6b, and 7a. Most of the



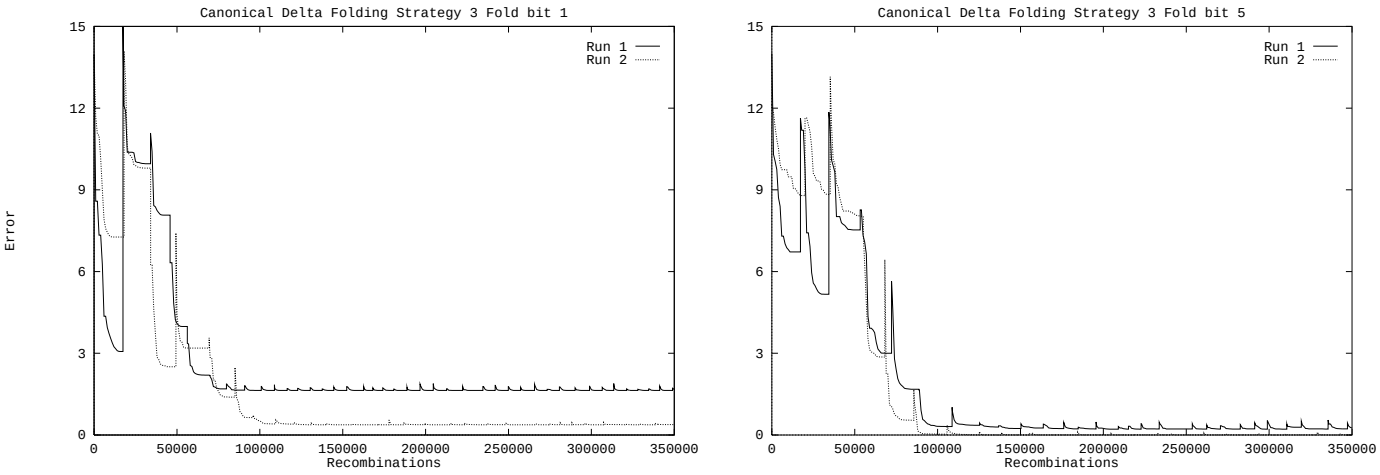
(a) (b)

Figure 6: Typical Delta Folding Searches Using Strategies 1 and 2.

canonical delta folding runs did not perform as well as delta coding. And in no case did the average performance of any canonical delta folding strategy match that of delta coding.

These different behaviors appear to be a result of the amount of locality in their subpartition sampling strategies. Strategy 1 provides for the sampling of the strings located adjacent in Hamming space to the interim solution and it's global complement. Empirical tests show that strings in this complementary subpartition often produce very poor solutions. Including the complementary subpartition in the reduced search space appears to impair the genetic algorithm's ability to find good solutions and results in a high degree of variance between searches.

Runs using strategies 2 and 3 provide for the sampling of subpartitions that are more local to the interim solution in Hamming space (Section 3.1) and are shown in figures 6b and 7a, respectively. These strategies typically produce better solutions and display less variance than searches using strategy 1.



(a) (b)

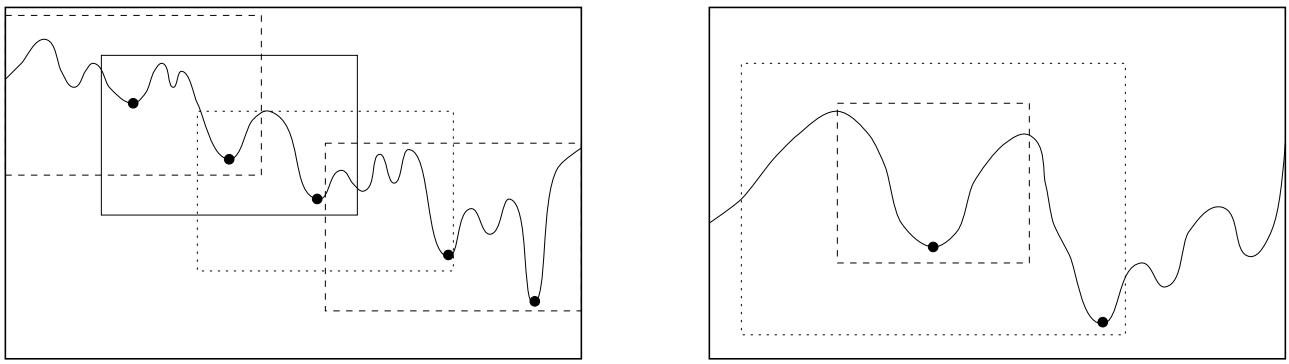
Figure 7: Typical Delta Folding Searches Using Strategy 3.

While it is not known how to choose an optimal folding bit for a particular problem when using a particular remapping strategy, the effects of applying a wide variety of fold bits combined with different sampling strategies were examined experimentally. The searches in Figure 7a employ a scheme using the first bit in each parameter for folding the hyperspace. The experiments in Figure 7b employ the fifth bit in each parameter for folding. While these runs are only representative of the many fold bit schemes that were tested, some schemes consistently produced better solutions than others. Also, given a fixed strategy, the choice of the fold bit can impact the variance (in terms of the best solution found) across multiple runs.

4.2 Exploiting Locality With A Moving Subpartition Window

Another conspicuous behavior of canonical delta folding is that *after the search has focused on a particular area of hyperspace the search makes no new progress* in spite of the re-initializations and remappings. This behavior is not observed in the delta coding runs. Delta coding continues to make progress toward a better solution over the length of the run. The computational behavior of delta coding and canonical delta folding was studied to try to isolate the cause of these differences.

Delta coding remaps and resamples subpartitions in the search space with each new iteration by sliding a window in numeric space (Figure 8a). This allows the algorithm to sample new points based on each new interim solution. Convergence to a new interim solution for each iteration enables the search to "crawl" along the error surface in search of new solutions. If the same solution is converged upon consecutively (for example, a local minima in a basin of attraction larger than the current sampling window, as in Figure 8b) then the re-expansion mechanism will enlarge the sampling window. This allows the search to escape the attracting basin (Figure 8b) and continue searching for new problem solutions.



(a) (b)

Figure 8: Side Affects of the Delta Coding Focus Mechanisms.

Canonical delta folding samples subpartitions based on adjacency in Hamming space, which is generally not equivalent to adjacency in numeric space. By constructing subpartitions of hyperspace representing numerically adjacent points, delta coding is able to relocate the reduced hypercube in the original search space. One failure of canonical delta folding which we have identified is that once the hypercube has been reduced, the more significant bits in the represen-

tation are fixed and the reduced hypercube cannot be relocated within the original search space. Consider strategy 1 in section 3.1. Note that remapping occurs only in the reduced hypercube; the same "pad bits" are always added as prefixes to map the reduced delta strings back to the canonical ordering of the full space. After exclusive-or is applied to the padded strings, the resulting prefixes can only match the prefix of the interim solution or its complement.

This means that the subpartition sampling window produced by strategy 1 cannot move, and that in order to sample a different subpartition of the space, the algorithm must converge to the same interim solution so that the search window can be re-expanded. However, *strategy 1 makes it difficult to converge to the same solution on consecutive iterations.* This is because the reduced search space *is* remapped at each iteration with respect to the new interim solution and the resulting hyperplane competitions can change dramatically at each iteration. The changing hyperplane competitions make it unlikely that the search will converge to exactly the same point twice in a row (although the same solution may be revisited over the course of the search many times). Thus, re-expansion is never triggered and the search effectively stagnates.

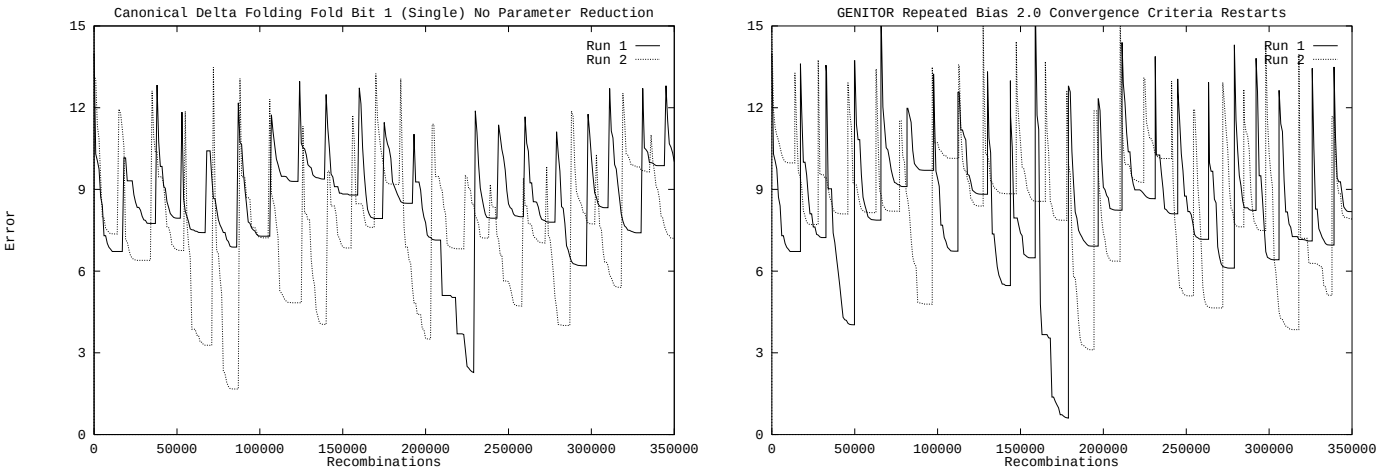
Strategies 2 and 3 have only slightly greater freedom than strategy 1. These strategies include limited information about a hypercube that is 1 dimension (assuming a single fold bit is used) larger than the current subpartition window. That is because the subpartition around the "local complement" is drawn from this larger subspace. The subpartitions sampled by the reduced hypercube can change to a limited degree, but only within the fixed subspace that is 1 dimension larger than the current reduced hypercube. In other words, the least significant pad bit can change values depending on the value of the interim solution, but all other pad bits are fixed.

4.3 Hyperspace Remapping Versus Sampling Bias

Figure 6a and 6b suggest that it is possible to determine that one subpartition sampling strategy produces better results than another after several iterations of remapping the search space. But a more basic question is whether one can evaluate the "goodness" of any particular remapping of hyperspace from one iteration to the next. Note that a subpartition sampling strategy is different from a remapping of hyperspace. To remove the effects of subpartition sampling from hyperspace remapping, the following tests were done without reducing the search space.

Figure 9a illustrates the behavior of an algorithm where the original search space is remapped at *each iteration* by moving the interim solution to the origin (using $\oplus$) and folding the space using C_{f1}. No subpartition sampling strategies are employed. Figure 9b shows the behavior of GENITOR using the same restart and convergence parameters, but no remapping mechanisms are applied. If remapping the space significantly affects the trajectory of the search, we would expect remapping to produce more variation in performance *as measured at the end of each iteration.* Analogously, we would expect the searches using GENITOR without remapping to display less variance.

The remapping algorithms were tested against the non-remapping algorithm using numerous population sizes for several different fold bits. No significant difference could be measured using an ANOVA test between the variance in performance associated with runs that remap the space



(a) (b)

Figure 9: Canonical Delta Folding(a) and Iterated GENITOR(b); No Search Space Reduction.

versus runs that did not. This suggests that the effects of remapping on performance cannot be distinguished from the effects of sampling bias associated with restarting the search. A good run may be due to remapping the space, or it may merely be due to fortuitous population sampling.

Another relevant observation is that both algorithms display "regression toward the mean." This implies that particularly good runs are the exception and that subsequent runs typically produce results that are closer to the mean. Thus even if one *could* identify a relatively good mapping of hyperspace, most subsequent mappings (if chosen randomly) would be inferior.

5 Conclusions

Analyzing the computational behavior of delta coding and canonical delta folding should lead to a better understanding of other algorithms that reduce the search space such as DPE (Schraudolph and Belew, 1990) and ARGOT (Shaefer, 1987). The results shown here should be interpreted with respect to the specific problem used in this study. Nevertheless, the limitations identified in canonical delta folding are general in nature and independent of the problem environment.

Locality in the subpartition sampling mechanism used to define the reduced hypercube appears to play a key role in performance. Strategies that use more local forms of subpartition sampling yield the best solutions and result in the least variance between runs. At the same time, it appears critical that this locality not be restricted to a fixed subpartition sampling window. The local search window used by delta coding allows the algorithm to exploit local optima while still allowing the algorithm to move to other portions of the search space. Canonical delta folding does not appear to exploit local information as well. This may be due to the fact that remapping causes the local view of the fitness landscape to constantly change.

Another phenomenon resulting from the subpartition sampling of Hamming space used by canonical delta folding is the effective stalling of the search. Unlike the numeric sampling strategy of delta coding, the subpartition sampling mechanisms of canonical delta folding fail to sample new partitions of the space after converging to a new interim solution. Failure to include

new sample points from the original search space in the construction of the reduced hypercube prevents the algorithm from discovering better solutions. This problem must be corrected in future versions of canonical delta folding. This may be a simple matter of finding a better subpartition sampling strategy or changing the mechanism detecting when search has stalled.

The subpartition sampling strategy used by delta coding and it's flexibility in relocating it's sampling window in hyperspace may be key to consistently finding better solutions quickly for this problem, in spite of the less general method of remapping. And even though the solutions may not be the global optimum, delta coding locates local optima that are competitive with the known optimal solutions. This may be sufficient for many optimization problems.

Experiments show that the choice of fold bit can affect the outcome of the search. While the use of some fold bits only occasionally result in good solutions, the use of other fold bits produce good results with little variance between runs. There is currently no evidence, however, that a particular fold bit may work well across all subpartition sampling strategies for this particular problem. This indicates that there is some interaction between subpartition sampling strategies and the remapping strategies that are employed in canonical delta folding. Additionally, the remapping of the search space by the fold bit when subpartition sampling strategies are not used seems to have little effect on the search results when compared to searches where remapping is not employed.

While the data indicates that the particular mappings provided by the choice of the fold bit may affect the results of the search, the evidence also suggests that it may be difficult to identify good mappings of the space. Certainly it appears to be difficult to identify good mappings by looking at performance from one iteration to the next. This is due in part to the sampling bias inherent in the randomly initialized populations used at the start of each new iteration. If the populations did not contain any sampling bias one would expect that all of the different mappings explored by canonical delta folding mechanisms would result in a much higher variance than the single mapping encoded for the basic GENITOR algorithm.

References

[1] Battle, D. and Vose, M. (1990) "Isomorphisms of Genetic Algorithms." In, *Foundations of Genetic Algorithms.* G. Rawlins, ed. Morgan Kaufmann.

[2] Liepins, G. and Vose, M. (1990) "Representation Issues in Genetic Algorithms." *Journal of Experimental and Theoretical Artificial Intelligence*, 2:101-115.

[3] Mathias, K. and Whitley, D., (1993) "Remapping Hyperspace During Genetic Search: Canonical Delta Folding." *Foundations of Genetic Algorithms 2.* D. Whitley, ed. Morgan Kaufmann.

[4] Schraudolph, N., Belew, R., (1990) "Dynamic Parameter Encoding for Genetic Algorithms." *CSE Technical Report* #CS 90-175.

[5] Shaefer, C., (1987) "The ARGOT Strategy: Adaptive Representative Genetic Optimizer Technique." *Genetic Algorithms and their Applications: Proc. of the* 2^{nd} *Int'l. Conf.* J. Greffenstette, ed.

[6] Whitley, D. (1989) "The GENITOR Algorithms and Selective Pressure: Why Rank Based Allocation of Reproductive Trials is Best." *Proc. of the* 3^{rd} *Int'l. Conf. on Genetic Algorithms.* Morgan Kaufmann.

[7] Whitley, D., Mathias, K., Fitzhorn, P. (1991) "Delta Coding: An Iterative Search Strategy for Genetic Algorithms." *Proc. of the* 4^{th} *Int'l. Conf. on Genetic Algorithms.* Washington, D.C., Morgan Kaufmann.