Transforming the Search Space with Gray Coding

Keith E. Mathias† and L. Darrell Whitley †

Abstract — **Genetic algorithm test functions have typically been designed with properties in numeric space that make it difficult to locate the optimal solution using traditional optimization techniques. The use of Gray coding has been found to enhance the performance of genetic search in some cases. However, Gray coding produces a different function mapping that may have fewer local optima and different relative hyperplane relationships. Therefore, inferences about a function will not necessarily hold when transformed to another search space. In fact, empirical results indicate that some genetic algorithm test functions are significantly altered by Gray coding such that local optimization methods often perform better than genetic algorithms.**

I. Introduction

Genetic algorithm test functions have typically been designed with properties that make it difficult to locate the optimal solution using traditional optimization techniques. These properties include non-linearity, multiple local optima, noise, and discontinuity [1]. Genetic algorithms globally sample the search space and are thought to be less sensitive to these properties. Thus, genetic algorithms are expected to perform better than local optimization methods on many of these test functions [3].

The use of Gray coding to transform functions found in standard genetic algorithm test suites [1, 9] has been shown to enhance the performance of genetic search in some cases [6, 10]. As a result, genetic algorithm performance comparisons are often accomplished using Gray coding [2, 4, 8, 10, 11]. However, Gray coding transforms a function so that it represents a different search space. Gray coding binary space not only eliminates Hamming cliffs but also has the potential to significantly alter the number of local optima in the search space. It also produces different relative hyperplane relationships, changing the schemata competitions in genetic search.

Evidence provided here shows that assumptions concerning the performance of local optimization methods on common genetic algorithm test functions do not apply when Gray coding is used. In fact, Gray coding enhances the performance of local optimization techniques on most of the DeJong test suite functions [1], as well as, the Rastrigin, Schwefel and Griewank functions [9]. In some cases the use of Gray coding results in a function mapping that some local search techniques are able to optimize faster than the genetic algorithm. Thus, performance comparisons of local and global search methods should be carried out using the same function mapping for all algorithms.

†Department of Computer Science, Colorado State University; Fort Collins, CO 80523. (mathiask, whitley@cs.colostate.edu)

This research was supported in part by a grant from the Colorado Advanced Software Institute.

II. Gray Coding the Function Space

Gray coding is a general method for transforming a function mapping such that the binary representations of consecutive numerical values differ by a single bit. A Gray coding algorithm produces a symmetric, one-to-one mapping between the binary and Gray coded function spaces and represents a non-linear transformation. It can be shown that there are at least $2^l(l!)$ Gray codes for a binary space represented by strings of length l. However *the* Gray code [5] algorithm most common in genetic algorithm testing [2, 4, 8, 10, 11] and used for these experiments is:

```
gray[0] = binary[0]
k = 1
WHILE (k < string_length)
{
   IF (binary[k-1] == 0) THEN  gray[k] = binary[k]
   ELSE  gray[k] = COMPLEMENT(binary[k]);
   k = k + 1
}
```

Binary strings can also be transformed into their Gray coded representation (for this Gray code) via matrix multiplication using an $l \times l$ matrix of the form:

- 1's along the main diagonal
- 1's along the upper/minor diagonal
- 0's everywhere else

where l is the string length and all addition is performed *mod* 2. Figure 1 lists the matrices that result in this Gray coding for strings of length 3, 4, and 5 bits. The matrices that produce $l! - 1$ of the other canonic Gray codes (i.e. the string of all 0's maps to the string of all 0's) can be constructed by reordering the columns of the appropriate matrix. It should also be noted that converting a Gray coded string to it's binary representation (de-Graying) can also be done via matrix multiplication.

$$\begin{vmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{vmatrix} \qquad \begin{vmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{vmatrix} \qquad \begin{vmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{vmatrix}$$

Figure 1: Example Gray Code Transformation Matrices.

Gray coding is known to eliminate Hamming cliffs that exist in binary function spaces. A Hamming cliff occurs when two consecutive numbers have complementary binary representations. For example, the binary representations for the numbers 7 and 8 are complements of each

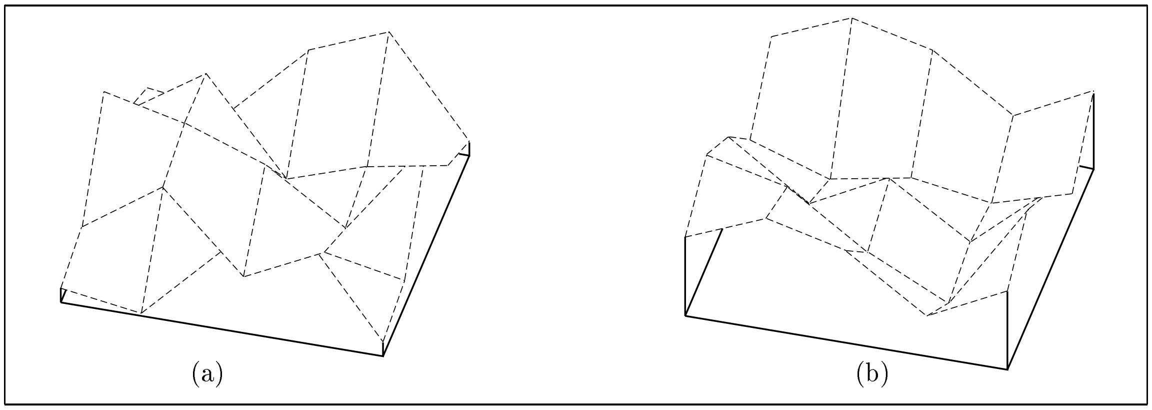


Figure 2: Three Dimensional View of Binary and Gray Coded Hamming Space for a Four Bit Version of the Griewank Function. The Gray coded space (b) is noticeably simpler than the binary space (a).

other (i.e. 0111 and 1000). Hamming cliffs often account for the inability of optimization methods that use binary encodings to locate the optimal solution of a function. The elimination of Hamming cliffs is a result of the more general property that Gray coding transforms complementary binary strings into representations that are adjacent in Gray Hamming space. For example, the Gray coded representations for the binary strings 000...000 and 111...111 are 000...000 and 100...000. This relationship can be generalized for any complementary pair of strings in binary space by applying exclusive-or ($\oplus$) over the strings in the canonic example [7]. Thus, two local optima located at complementary string positions in binary space will be in the same attraction basin in Gray coded space.

Gray coding has the potential to increase or decrease the number of local optima for any function. Table 1 shows the number of local optima in binary and Gray coded Hamming space for 1- and 2-dimensional versions of the Rastrigin (F6), Schwefel (F7) and Griewank (F8) functions [9], using 10 bit encodings. While there are fewer local optima in Gray coded Hamming space than in binary space for F6 and F7, F8 has approximately the same number in both spaces.

Function	F6	F6	F7	F7	F8	F8
Dimension	1	2	1	2	1	2
Binary	19	361	12	144	18	627
Gray	5	25	5	25	22	639

Table 1: Local Optima in Hamming Space.

Table 1 also shows how local optima grow with respect to multi-dimensional functions. The number of local optima in multi-dimensional functions that are a linear combination of a non-linear function (i.e. F6/F7), grow according to l^d, where l is the number of local optima in the 1-dimensional function and d is the dimension. Therefore, if more local optima exist in binary space for the 1-dimensional function than in Gray space, then as the dimensionality increases the number of local optima in binary space grow exponentially faster than in Gray space. Thus Gray coding solves a simpler function mapping.

This behavior is not observed for F8, where parameter interactions are multiplicative. As the dimensionality of F8 increases the number of local optima increase more rapidly than predicted by the l^d growth expression.

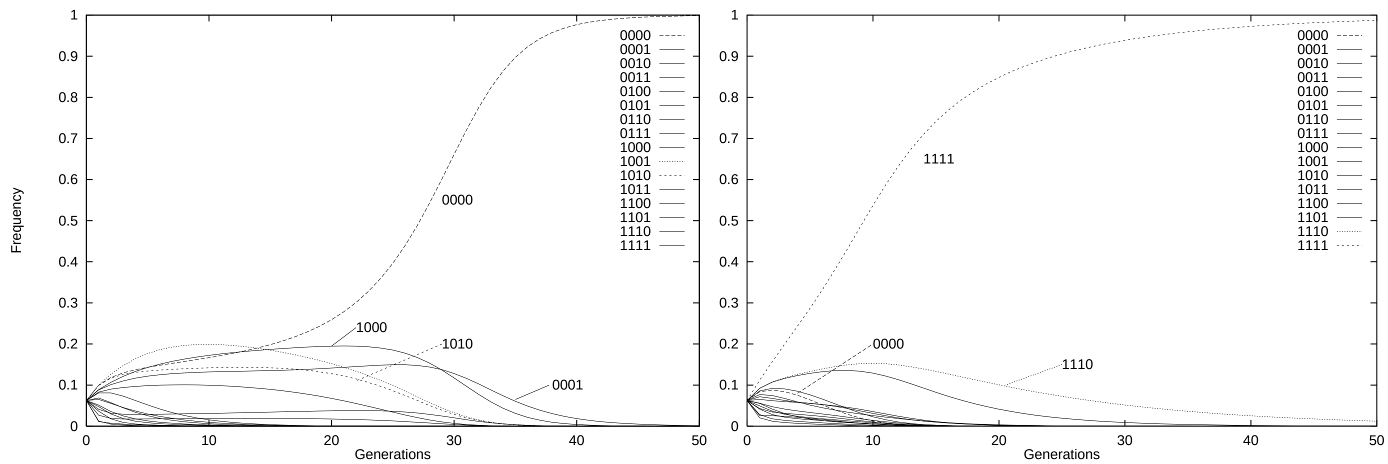
Figure 2 provides a 3-dimensional representation of the binary (a) and Gray coded (b) Hamming spaces for a four bit Griewank function. Fitness is represented as elevation and Hamming space is arranged according to the pattern:

1111	1110	1010	1011	1111
1101	1100	1000	1001	1101
0101	0100	0000	0001	0101
0111	0110	0010	0011	0111
1111	1110	1010	1011	1111

The neighbors to the north, south, east and west of each string differ by 1 bit. The points along the edges are duplicated so that neighborhoods are completed with minimal wrap around. Local minima in binary Hamming space are located at the points 0001, 0010, and 1110. The global minimum is located at 1000. Local minima in Gray coded Hamming space are located at the points 0011 and 1001. The global minimum is represented by the string 1100.

III. Gray Coding the Genetic Search Space

While Gray coding has been shown to enhance the performance of genetic algorithms in some instances [2, 6, 10] it can be demonstrated that there is no reason to expect that this behavior will hold for any arbitrary function. Every function mapping that exists in binary space also exists in Gray coded space. Therefore, assume some function f exists in binary space which is difficult with respect to the genetic algorithm. This same function may be much simpler for the genetic algorithm when Gray coded. However, now assume that same function f, is a function in Gray coded space which must be de-Grayed to find the binary coding of the function. Function f now represents a function that is hard for the genetic algorithm in Gray space. Thus, every function existing in binary space that is difficult for the genetic algorithm represents a function in Gray space that is difficult for the genetic algorithm.



(a) (b)

Figure 3: String Competitions for Binary and Gray Coding of Four Bit Problem.

Gray coding a function remaps the hyperplane relationships within the search space. This alters the schemata competitions that occur during genetic search. This is illustrated using the function given in Table 2 and the executable model of the simple genetic algorithm (SGA) [3] developed by Whitley [12]. This executable model allows us to track the representation of the strings in the population over time for a given problem. The model assumes an infinite population and does not model the use of mutation. The model also requires the fitness of all points in the search space. It then calculates the proportional representation for every string in the population each generation based on each string's relative fitness and current representation in the population.

Binary	Gray	Fitness	Binary	Gray	Fitness
0000	0000	30	1000	1100	18
0001	0001	23	1001	1101	24
0010	0011	8	1010	1111	28
0011	0010	4	1011	1110	26
0100	0110	12	1100	1010	16
0101	0111	20	1101	1011	22
0110	0101	10	1110	1001	14
0111	0100	2	1111	1000	0

Table 2: Four Bit Function.

Based on the executable model output for the binary search space, Figure 3a shows that the proportional representation of the optimal string 0000 monotonically increases over time. By generation 20 the SGA has begun to converge on the optimal string and by generation 40 almost no other strings are represented in the population. However, the executable model output shown in Figure 3b shows that when the Gray coding representation of the problem is used, the SGA rapidly converges on the string 1111 (1010, binary). This behavior is due to the remapping of the original hyperplane relationships, producing schemata competitions that mislead the SGA. This example not only illustrates how Gray coding remaps hyperspace but also how it can produce a function that is more difficult for the SGA to solve.

IV. Empirical Performance Results

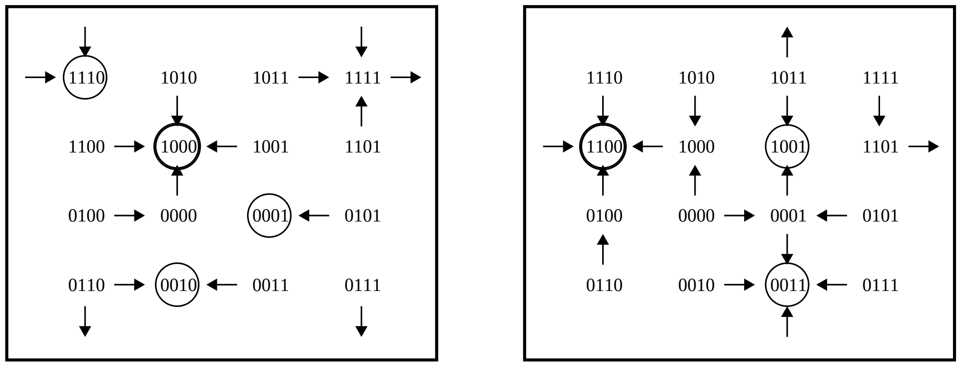
Here, we first examine the effects of Gray coding with respect to steepest descent search. The steepest descent algorithm used here begins at a random point in Hamming space and then evaluates the fitness of its l nearest neighbors, where l is the length of the string. The algorithm moves to the neighbor with the smallest fitness value if it is less than or equal to the fitness of the current point. This process is repeated until a better solution cannot be found. If the optimum solution is not reached a new random starting point is generated.

Steepest descent search was applied to the DeJong test suite (F1-F5) [1] and the F6, F7 and F8 test functions 30 independent times. A maximum of 1000 random starting points was allowed for each run. The results are shown in Table 3. *Number solved* indicates the number of independent runs in which the optimal solution was located. *Average restarts* indicates the number of random start points necessary to locate the optimum solution, averaged over those runs that were successful. *Average best* reflects the best solution found averaged over all 30 runs.

The results in Table 3 can also be used to gain some indication as to the number of restarts necessary to find the optimal solution. This can be accomplished by entering the table values for the number solved (Slv), average restarts (Avg), number of independent runs (N), and maximum restarts allowed in an independent run (Max) into the following equation:

$$\frac{(Slv * Avg) + Max(N - Slv)}{Slv}$$

Table 3 indicates that Gray coding significantly enhances the performance of steepest ascent search for most of the test functions here. However, this behavior is particularly surprising for F8. Based on the growth rate of



(a) (b)

Figure 4: Binary and Gray Coded Hamming Space with Respect to Steepest Ascent.

Coding	Func.(Dimension)	F1 (3)	F2 (2)	F3 (5)	F4 (30)	F5 (2)	F6 (20)	F7 (10)	F7 (20)	F8 (10)	F8 (20)
Binary	Number Solved	30	9	30	3	30	0	0	0	2	0
	Average Restarts	8.3	483.7	79.7	610.0	17.4				468.0	
	Average Best		0.000009		-1.68			-4014.6		0.069	0.145
Gray	Number Solved	30	15	28	0	30	0	3	0	30	30
	Average Restarts	1	466.9	264.6		2.2		514.0		37.5	5.5
	Average Best		0.000001	-29.93	-0.744		17.39	-4032.2	-7626.6		

Table 3: Steepest Ascent Performance. All comparisons are over 30 independent runs.

local minima for F8, given in Table 1, the binary and Gray codings should prove equally difficult. Further analysis of smaller versions of F8 suggests that the *percentage* of points in the attraction basin of the global optimum increases dramatically when Gray coding is applied.

This phenomenon can be illustrated by using directional descent diagrams to represent the 4 bit Griewank function shown in Figure 2. Directional descent diagrams indicate the path followed by steepest ascent search with arrows. There are 4 attraction basins in binary Hamming space (Figure 4a) with optima at 0001, 0010, 1110, and 1000. There are 2, 3, 6, and 6 points in each respective attraction basin. The attraction basins in Gray coded Hamming space (Figure 4b) whose optima are located at 0011, 1001, and 1100 have sizes 7, 5 and 9 respectively. The global attraction basin in Gray space includes 43% of the points as compared to 35% in binary space. Analysis based on full 10 bit versions of the Griewank function, indicate that the relative size of the attraction basin for the global optimum in Gray space increases more dramatically as the dimensionality is increased.

Next, we compared the performance of the CHC genetic algorithm [2] with a mutation driven stochastic hill-climber. We used CHC for comparison purposes because of its robustness and superior performances in other comparative studies [2, 8]. All CHC tests employed a population of size 50 and a cataclysmic mutation rate of 35%.

The stochastic hill climber began by randomly generating a binary string. Then search was performed by applying mutation to that string. The changes produced were kept whenever the fitness of the resulting string was better than or equal to the string before mutation. This process was repeated until the optimal solution was found or until some maximum number of trials were executed. We tested mutation rates ranging from 0.002 to 0.09 where the *mutation rate* is the probability that a bit is complemented. Only the best results are reported here.

Performance comparisons using a binary encoding of the test functions are recorded in the top half of Table 4. The Gray coding results are recorded in the bottom half of the table. Each algorithm was run 30 times for each function. *Average trials* indicates the number of function evaluations necessary to locate the optimum solution, averaged over those runs that were successful. For those functions where the algorithm was not successful in consistently locating the optimal solution, the maximum number of function evaluations allowed for a run (i.e. *maximum trials*) and the *average best* is also listed. All other labels are the same as described earlier for Table 3.

The results in Table 4 show that when a binary coding of the test functions is used, CHC performs significantly better than the stochastic hill-climber with a single exception (F3). The hill-climber fails to consistently locate the global optimum for all but 1 of the 10 test functions. Thus, the inferences concerning relative difficulty of these test functions with respect to local search generally seem to hold when using a binary encoding.

However, when Gray coding is used to transform the search space of these test functions, the performance of both algorithms is dramatically improved. Surprisingly,

Algorithm	Problem (Dim)	F1 (3)	F2 (2)	F3 (5)	F4 (30)	F5 (2)	F6 (20)	F7 (10)	F7 (20)	F8 (10)	F8 (20)
	Total Bits	30	24	50	240	34	200	100	200	100	200
BINARY ENCODING											
Stochastic Hill Climber	Percent Solved	20%	23%	100%	20%	40%	0%	40%	0%	0%	0%
	Average Trials	384	27672	544	62284	884		346667			
	Maximum Trials	200000	100000		100000	50000	500000	500000	500000	500000	500000
	Average Best	0.0001	0.0002		-1.893	1.16	6.71	-4170.72	-7838.44	0.182	0.3398
	Mutation Rate	0.05	0.09	0.04	0.03	0.08	0.03	0.05	0.03	0.03	0.01
CHC	Percent Solved	100%	100%	100%	100%	100%	00%	100%	100%	23%	00%
	Average Trials	56892	37737	687	14715	4443		15230	57304	345242	
	Maximum Trials						500000			500000	500000
	Average Best						0.18			0.038	0.271
GRAY ENCODING											
Stochastic Hill Climber	Percent Solved	100%	100%	100%	30%	100%	100%	100%	7%	3%	3%
	Average Trials	507	19591	767	58067	481	104844	115267	455001	227701	60001
	Maximum Trials				100000				500000	500000	500000
	Average Best				-2.038				-8091.31	0.099	0.093
	Mutation Rate	0.04	0.15	0.04	0.04	0.07	0.02	0.05	0.03	0.02	0.03
CHC	Percent Solved	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	Average Trials	1126	9455	1265	16335	733	158839	9803	17123	51015	50509

Table 4: Comparative Performance Using Binary and Gray Encodings.

the stochastic hill-climber solves all of the test functions and all but 4 are solved consistently. Perhaps even more important is the fact that when Gray coding is used, this local search method was able to perform better than CHC on almost half of these test functions. This indicates that the properties which make these test problems challenging for optimization methods that use a binary encoding of the search space are lost when the functions are transformed by Gray coding.

V. Adjacency Analysis

Gray coding has been shown here to compromise the difficulty of certain test functions. Nevertheless, it would be useful to find some measure indicating the degree to which a coding preserved properties in numeric space. One such measure might be the pair-wise distance of the points in the search space. This can be approached by constructing a Toeplitz matrix for numeric space ($Numeric$) such that each element in the lower triangle is: $n_{i,j} = |(i-j)|$, where i and j are the consecutive numerical value labels.

A similar matrix can be constructed for binary space ($Binary$) such that each element in the lower triangle is: $b_{i,j} = H_d(b_i, b_j)$, where b_i and b_j are the binary representations of consecutive numerical values and H_d determines the Hamming distance between two elements. The Hamming distance matrix for the Gray coded space ($Gray$) can be constructed in the same way. All elements in the major diagonal for these matrices are 0. Additionally, the $Gray$ and $Numeric$ matrices have 1's along the minor diagonal and 2's along the next successive minor diagonal.

$$\left|\begin{array}{cccccccccccccccc}
0 \\
0 & 0 \\
1 & 1 & 0 \\
1 & 1 & 0 & 0 \\
3 & 1 & 0 & 2 & 0 \\
3 & 3 & 0 & 0 & 0 & 0 \\
4 & 2 & 3 & 1 & 1 & 1 & 0 \\
4 & 4 & 3 & 3 & 1 & 1 & 0 & 0 \\
7 & 5 & 4 & 2 & 2 & 0 & 1 & 3 & 0 \\
7 & 7 & 4 & 4 & 2 & 2 & 1 & 1 & 0 & 0 \\
8 & 6 & 7 & 5 & 3 & 1 & 2 & 0 & 1 & 1 & 0 \\
8 & 8 & 7 & 7 & 3 & 3 & 2 & 2 & 1 & 1 & 0 & 0 \\
10 & 8 & 7 & 5 & 7 & 5 & 4 & 2 & 3 & 1 & 0 & 2 & 0 \\
10 & 10 & 7 & 7 & 7 & 7 & 4 & 4 & 3 & 3 & 0 & 0 & 0 & 0 \\
11 & 9 & 10 & 8 & 8 & 6 & 7 & 5 & 4 & 2 & 3 & 1 & 1 & 1 & 0 \\
11 & 11 & 10 & 10 & 8 & 8 & 7 & 7 & 4 & 4 & 3 & 3 & 1 & 1 & 0 & 0
\end{array}\right|
\qquad
\left|\begin{array}{cccccccccccccccc}
0 \\
0 & 0 \\
0 & 0 & 0 \\
2 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 \\
2 & 2 & 2 & 0 & 0 & 0 \\
4 & 4 & 2 & 0 & 0 & 0 & 0 \\
6 & 4 & 2 & 2 & 2 & 0 & 0 & 0 \\
6 & 4 & 2 & 2 & 2 & 0 & 0 & 0 & 0 \\
6 & 6 & 4 & 2 & 2 & 2 & 2 & 0 & 0 & 0 \\
6 & 6 & 6 & 4 & 4 & 4 & 2 & 0 & 0 & 0 & 0 \\
8 & 6 & 6 & 6 & 6 & 4 & 2 & 2 & 2 & 0 & 0 & 0 \\
10 & 8 & 8 & 8 & 6 & 4 & 2 & 2 & 2 & 0 & 0 & 0 & 0 \\
10 & 10 & 10 & 8 & 6 & 6 & 4 & 2 & 2 & 2 & 2 & 0 & 0 & 0 \\
12 & 12 & 10 & 8 & 6 & 6 & 6 & 4 & 4 & 4 & 2 & 0 & 0 & 0 & 0 \\
14 & 12 & 10 & 10 & 8 & 6 & 6 & 6 & 6 & 4 & 2 & 2 & 2 & 0 & 0 & 0
\end{array}\right|$$

(Binary) (Gray)

Figure 5: Four Bit Binary and Gray Distance Matrices: Relative to the Toeplitz Matrix for Numeric Space.

One method for comparing the relative similarity of the matrices is to calculate the Lx Norms for the *Binary* and *Gray* matrices with respect to the *Numeric* matrix. This is done according to the expression:

$$L(x)Norm = \sqrt[x]{\sum_{i,j}(|n_{i,j} - b_{i,j}|)^x}$$

where $n_{i,j}$ are the elements of the *Numeric* matrix and $b_{i,j}$ are the elements of the *Binary* matrix. This can be repeated using the elements of the *Gray* matrix (i.e. $g_{i,j}$) in place of the *Binary* elements. The absolute value of the pair-wise differences (i.e. $|n_{i,j} - b_{i,j}|$) used in the 4 bit $L(x)$ *Norm* computations for both the binary and Gray coded cases are shown in Figure 5.

The $L1$, $L2$, and $L3$ Norms for 3, 4, and 5 bit spaces are shown in Table 5. The $L1$ Norms are smaller for Gray coded spaces than the corresponding binary spaces. However, the $L3$ Norms for Gray coded spaces are larger than those for corresponding binary spaces and grow more quickly. A count of the non-zero elements in the matrices in Figure 5 shows that 85 elements have been remapped for the 4 bit Gray coded space while 104 elements have been remapped in the corresponding binary space. Thus, dividing the $L1$ Norms by the number of elements that are remapped in the corresponding matrix yields 4.35 for the binary space and 4.99 for the Gray space in the 4 bit example.

Another method for measuring the degree of similarity between the *Numeric* matrix and the *Binary*/*Gray* matrices is to sum the elements of the matrices in Figure 5, weighted by the square of the distance of the element from the diagonal then divide by the sum of all elements as in:

$$WtDiag = \frac{\sum_{i,j} z_{i,j}(i-j)^2}{\sum_{i,j} z_{i,j}}$$

where $z_{i,j}$ are the elements of the matrix being operated on. The results of this computation for 3, 4, and 5 bit spaces also appear in Table 5. This measure emphasizes that the elements in Gray space that are being adjusted the most are those furthest from the diagonal. Therefore the *WtDiag* Norm grows faster for the Gray difference matrix than the corresponding binary difference matrix. These metrics indicate that fewer points are remapped in Gray space than in the corresponding binary space but the adjustments being made in Gray space are generally larger. The binary encoding however, generally makes smaller changes but does so more globally.

Bits	Matrix	L1 Norm	L2 Norm	L3 Norm	WtDiag Norm
3	Binary	44.00	10.95	7.24	21.18
	Gray	36.00	10.95	7.83	26.78
4	Binary	452.00	53.81	28.16	85.27
	Gray	424.00	53.81	28.95	92.35
5	Binary	4256.00	245.34	100.84	332.08
	Gray	4176.00	245.34	101.88	340.99

Table 5: Distance Matrix Norms.

VI. Conclusions

Gray coding transforms a test function so that it represents a new function mapping. The new function mapping may have a different number of local optima, different relative attraction basin sizes and/or different hyperplane competitions. Therefore, inferences concerning the difficulty of a test function with respect to an algorithm no longer hold after it has been transformed. Consequently, all algorithm performance comparisons must be executed using the same search space encoding.

Theoretically, there is no reason to expect that Gray coding an *arbitrary* function will make it any easier to locate the optimum solution via genetic search. Nevertheless, the Gray coding of these standard test functions has a significant impact on the performance of hill climbing mechanisms. This suggests that the continued use of Gray coding in genetic algorithm comparisons may result in the perception that those genetic algorithms with strong hill climbing components are the superior algorithms.

Finally, evidence presented here indicates that the complexity of current multi-dimensional test suite functions does not grow as expected when Gray coding is used. In fact, as the dimensionality is increased some test functions, such as the Griewank function, actually become "easier" for many search methods. Thus, scalability testing of genetic algorithms involving the Gray coding of current test functions may be suspect. This may also indicate that new test functions which include *non-linear interactions between all parameters* should be developed.

Acknowledgements

Our thanks to Raja Das, John Dzubera, Scott Gordon and Sumit Sur for making critical suggestions and reading earlier drafts.

References

[1] Ken DeJong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, University of Michigan, Department of Computer and Communication Sciences, Ann Arbor, Michigan, 1975.

[2] Larry Eshelman. The CHC Adaptive Search Algorithm. How to Have Safe Search When Engaging in Nontraditional Genetic Recombination. In G. Rawlins, editor, *FOGA -1*, pages 265–283. Morgan Kaufmann, 1991.

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

[4] V. Gordon and D. Whitley. Serial and Parallel Genetic Algorithms as Function Optimizers. In Stephanie Forrest, editor, *Proc. of the 5th Int'l. Conf. on GAs*, pages 177–183. Morgan Kauffman, 1993.

[5] Richard Hamming. *Coding and Information Theory.* Prentic-Hall, Inc., Englewood Cliffs, NJ, 1980.

[6] R. B. Hollstien. *Artificial Genetic Adaptation in Computer Control Systems.* PhD thesis, University of Michigan, Department of Computer and Communication Sciences Ann Arbor, Michigan, 1971.

[7] Keith E. Mathias and L. Darrell Whitley. Remapping Hyperspace During Genetic Search: Canonical Delta Folding. In L. Darrell Whitley, editor, *FOGA - 2*, pages 167–186. Morgan Kaufmann, 1993.

[8] Keith E. Mathias and L. Darrell Whitley. Changing Representations During Search: A Comparative Study of Delta Coding. *Journal of Evolutionary Computation*, 2(3):249–278, 1994.

[9] H. Mühlenbein, M. Schomisch, and J. Born. The Parallel Genetic Algorithm as Function Optimizer. In L. Booker and R. Belew, editors, *Proc. of the 4th Int'l. Conf. on GAs*, pages 271–278. Morgan Kauffman, 1991.

[10] 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, *Proc. of the 3rd Int'l. Conf. on GAs*, pages 51–60. Morgan Kauffman, 1989.

[11] N. Schraudolph and R. Belew. Dynamic Parameter Encoding for Genetic Algorithms. *Machine Learning*, 9:9.21, 1992.

[12] L. Darrell Whitley. An Executable Model of the Simple Genetic Algorithm. In L. Darrell Whitley, editor, *FOGA - 2*, pages 45–62. Morgan Kaufmann, 1993.