Representation, Search and Genetic Algorithms

Darrell Whitley **Soraya B. Rana**
Computer Science Department
Colorado State University
Fort Collins, Colorado 80523
email: {whitley,rana}@cs.colostate.edu

Abstract

Wolpert and Macready's No Free Lunch theorem proves that no search algorithm is better than any other over all possible discrete functions. The meaning of the No Free Lunch theorem has, however, been the subject of intense debate. We prove that for local neighborhood search on problems of bounded complexity, where complexity is measured in terms of number of basins of attraction in the search space a Gray coded representation is better than Binary in the sense that on average it induces fewer minima in a Hamming distance 1 search neighborhood.

Introduction

Wolpert and Macready's No Free Lunch theorem (Wolpert & Macready 1995) shows that the performance of no search algorithm is better than another when averaged over the set of all possible discrete functions. We explore the concept of No Free Lunch as it relates to representations of optimization and search problems that are coded as bit strings of length L. Bit representations of this type are used by traditional genetic algorithms as well as other local search methods, such as Random Bit Climbing (Davis 1991).

Of course, there are actually many possible bit representations: standard Binary encodings and standard Binary Reflective Gray code are the two most commonly used. But if we consider all bit representations over all possible functions, Binary is no better than Gray encoding. Every bit representation can be decoded as a Binary representation of a function f_1 or as the reflective Gray codings of a function f_2. The set of all possible Gray representations and the set of all Binary representations is just the set of all possible functions defined over bit strings; therefore, the sets of all Gray and all Binary representations are identical. Hence, there is No Free Lunch.

Despite No Free Lunch results, applications oriented researchers have argued for the use of Gray codes. It is well known that Gray codes remove Hamming cliffs. A Hamming cliff occurs when numbers that are adjacent integers have Binary representations that are complements: for example 3 and 4 (011 and 100). Also, it has been shown that a Gray code representation induces fewer minima for some common test functions (Mathias & Whitley 1994b). Practitioners have argued that real world applications have structure that bounds the complexity of the problems that we actually want to solve: No Free Lunch is no big deal. Yet the conjecture that No Free Lunch does not hold for problems of bounded complexity is largely unexplored.

We propose a measure of complexity that counts the number of local minima that exist in a nonbinary Defining neighborhood representation of the domain of the functions. Functions that have fewer minima are assumed to be less complex than functions with more minima. We prove that for local neighborhood search on problems of bounded complexity, where complexity is measured in terms of number of basins of attraction in the search space a Gray coded representation is better than Binary in the sense that on average it induces fewer minima in a Hamming distance 1 neighborhood.

The debate as to whether Gray coding is better than Binary representations has been a classic example of where theory and practice clash. The results in this paper bring theory and practice closer together and yields new insights into the role of representation as it impacts search. This also leads to insights about the relative merits of Binary versus Gray encodings for genetic algorithm representations. The empirical evidence indicates that genetic algorithms perform better on Gray coded representations than Binary representations for most common test functions (Caruana & Schaffer 1988) (Mathias & Whitley 1994a); in addition, the Gray coded representations induce fewer minima than the corresponding Binary representations of common test functions.

A Finite Set of Discrete Functions

Because we are only interested in the set of functions which can be represented as bit strings, we also restrict our attention to a special set of discretized functions whose domain are the integers 0 to $2^L - 1$ and hence can be represented using an L-bit representation; if a function has multiple input parameters, then each parameter, p, must map onto l_p bits and has a domain 0 to $2^{l_p} - 1$. If the natural domain of the function and its parameters is not these integers, we assume some auxiliary mapping exists onto this integer domain.

We also restrict our attention to functions that have as their range 1 to 2^L. For discrete functions with different ranges we assume that the values in the natural range of the functions can be sorted; ties can be broken in an arbitrary fashion. We then map the i^{th} element in the sorted natural range to the integer i.

This reduction of the domain and range creates a well defined finite set of functions, which we denote by F; f_j is a function in F. Let g be a function whose domain has already been mapped to the set of integers 0 to $2^L - 1$, but where the range is defined arbitrarily. We map g to a function f_i by ranking all possible outputs of g. Thus the range of all functions in F is limited to 1 to 2^L. For convenience, we sometimes talk about functions in F as if they are 1-dimensional functions with only 1 parameter coded using L bits, but all statements apply equally to N-dimensional functions with N parameters, $N \leq L$; we discuss N-dimensional functions when there is potential for confusion.

Now consider a local search operator with the following neighborhood structure. For all integers, $i \in [0, 2^L]$, and for all parameters, p, in the domain of function f_j the points $i - 1$ and $i + 1$ are neighbors; endpoints at 0 and $2^L - 1$ (or 2^{l_p} for parameter p) are also considered adjacent. We will refer to the neighborhood structure induced by this as the *Defining neighborhood*. Given a point in the search space, we use a local search operator that increments or decrements each input parameter by 1, accepting either the first improving move or the best improving move in this neighborhood. Without loss of generality we assume we are minimizing the function. If all points in the neighborhood are worse than the current solution, the current solution is a local minimum. Every local minimum in g is also a local minimum in f_j since we preserved the relative value of the strings in the mapping of g onto f_j. Also, the basins of attraction that lead to a local optimum are identical for both g and f_j. Furthermore, function g and f_j will be processed identically by any genetic algorithm that uses a rank-based selection scheme, including Tournament Selection and Truncation Selection.

Bit Representations

Typically, with a bit representation a local search operator searches a neighborhood of L bits for an improving move. We refer to this as the Hamming distance 1 neighborhood, or *Hamming neighborhood*. Thus, for an N dimensional function, the number of neighbors for any point in the space is more in the Hamming neighborhood (i.e., the Hamming distance 1 neighborhood) compared to the $2N$ points in the Defining neighborhood, provided that there is an least 1 parameter, p, that has an encoding using $l_p > 2$ bits. When $N = L$, each parameter takes on only 2 values and is coded with a single bit; in this case there is no difference between the Hamming neighborhood and the Defining neighborhood. Also in this case there is no difference in the Binary and Gray representation since the Binary coding and Gray coding of a bit string of length 1 are identical. If a parameter can take on 4 values and is coded using 2 bits, then the neighborhood *size* associated with the parameter is again the same: there are 2 neighbors in the Defining neighborhood and 2 neighbors in the Hamming neighborhood. However, the Gray codes and Binary codes are not the same: Binary maps the integer 2 and 3 to strings 10 and 11 respectively, while Gray maps the integers 2 and 3 to strings 11 and 10 respectively. If a parameter can take on more than 4 values, then the Hamming neighborhood is larger than the Defining neighborhood.

We will denote the set of all bit representations of functions in F as R; R is really all possible functions defined over bit strings of length L since R explicitly includes the string representation as well as the evaluations of those strings. When the parameters in the domain of every function in F are converted into their Binary representation, a one-to-one and onto mapping is created between F and R; Gray coding also induces a one-to-one and onto mapping between F and R, but the mapping is different. Thus, R isn't a Binary representation or a Gray representation. Instead, Binary and Gray representations are each just invertible mappings between F and R.

One might assume that there are fewer local minima in the Hamming neighborhood topology of an arbitrary bit representation of a function compared to the Defining neighborhood topology. But this isn't necessarily true; if a function is highly structured, then an arbitrary bit representation can destroy the structure and induce more local minima than exists in the Defining neighborhood. For common test functions the number of minima that exists under standard Gray and Binary representations is dramatically smaller than the average number of minima for an arbitrary representation (Rana & Whitley 1997).

The Family of Gray Codes

Usually "Gray code" refers to Standard Binary Reflective Gray code, which we also refer to as, reflective Gray code. Reflective Gray coding is just one of many possible representations that belong to the general family of Gray codes. A Gray code is any representation where integers that are adjacent also have bit representations that are adjacent; for any integer i the bit representation of i is Hamming distance 1 away from the bit representations of the integers $i+1$ and $i-1$. The addition and subtraction operators are mod 2^L (or 2^{l_p}), so that 0 and the maximum represented integer are adjacent as well.

Reflective Gray code is easy to compute: construct the standard Binary representation of any integer i. Apply logical "and" to the L-bit Binary representation and a duplicate of the Binary bit representation that has been shifted 1 bit to the right. The first L bits of the "and" operation is the reflective Gray code of i.

Another way to generate the reflective Gray code is by using a transformation matrix. There exists an $L \times L$ matrix G_L that transforms a string of length L from Binary to the reflective Gray representation. There also exists a matrix D_L that maps the Gray string back to its Binary representation. The matrices G_4 and D_4 (for strings of length 4) matrices are:

$$G_4 = \begin{vmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{vmatrix} \qquad D_4 = \begin{vmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

In higher dimensions the G_L matrix continues to have "1" bits along the diagonal and the upper minor diagonal, and D_L has "1" bits in the diagonal and the upper triangle. Given a bit vector, x, which is the Binary vector for integer i, $x^T G_L$ produces a Gray coding of x and integer i; $x^T D_n$ produces a DeGray coding of x, where x is an L-bit column vector. Addition operations during the matrix multiply are addition mod 2. Transformation matrices for other Gray codes can be produced by permuting the columns of G_L; other Gray codes can also be produced by first adding a constant Y to each integer, then generating the Binary representation and Graying using G_L or some permutation of G_L. This suggests that for strings of length L, on the order of $L2^L$ Gray codes may exist. The exact number of Gray codes that exist for strings of length L is an open question.

Gray Beats Binary: An Example

To find differences between Gray representations and Binary representations, we must partition the set of functions into special subsets. But we would like these subsets to be indicative of problem complexity. One way to do this is to partition the set of all functions according to the number of local optima that occur in the representation induced by the *Defining neighborhood* of the function. Actually we only partition the finite set of functions F, but this also partitions all functions that can be mapped onto F. Our assumption is that less complex functions have fewer local optima in the *Defining neighborhood* topology; more complex functions have more local optima.

We look at all functions in F that have Q or fewer local optima in their Defining neighborhood topology; we then ask, which induces fewer local optima in the resulting sets of bit representation, Gray or Binary? We next give a specific example where the Gray representation is better than Binary.

Consider functions in F with a 2 bit representation. For now assume the function has no duplicates in the range of outputs. We need only consider 1-dimensional functions where a single 2-bit parameter is taken as input. For 2 dimensional functions there are 2 1-bit parameters so that the Gray and Binary encodings are identical. Thus, in this case, if there are differences between Gray and Binary representations, they only exist for the 1-dimensional functions. By definition, the domain of F is $\{0,1,2,3\}$ and the range $\{1,2,3,4\}$.

Now consider the following two functions, f_a and f_b:

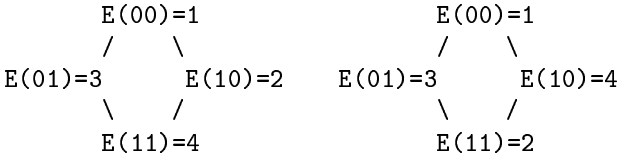
$$f_a(0)=1, f_a(1)=3, f_a(2)=2, f_a(3)=4$$

which we will denoted by $f_a = 1324$ for brevity.

$$f_b(0)=1, f_b(1)=3, f_b(2)=4, f_b(3)=2 \text{ or } f_b = 1342$$

Function $f_a = 1324$ has 2 optima in its Defining neighborhood topology at $f_a(0)=1$ and $f_a(2)=2$. Function $f_b = 1342$ has 1 optimum at $f_b(0)=1$.

The following graphs are the Hamming neighborhoods under Gray and Binary representations. However, the diagram does not distinguish between F_a and F_b or Gray and Binary. Let the representation on the left be $r_a \in R$ and the representation on the right be $r_b \in R$. Then r_a is the Binary code of f_a and r_b is the Gray code of f_a. But r_a is also the Gray code of f_b and r_b is the Binary code of f_b.



E is the evaluation function in R. Representation r_a has 1 minimum at 00 under the Hamming neighborhood search operator while r_b has 2 minima: 00 and 11.

Assume we are interested in functions with 1 minimum in the Defining neighborhood topology. Under this restriction, we are interested in f_b, but not f_a.

All 1-D functions with unique values that can be mapped onto a 2-bit encoding can be represented by permutations over 1 2 3 4. There are only 4! or 24 possible functions (permutations) over these values. Let all functions in F with 1 minimum be in class-1 and all functions in F with 2 minima be in class-2. We then show that Gray is better over the class-1 functions in F with 1 minimum in the sense that Gray induces fewer minima than Binary in the resulting Hamming neighborhoods. Since there is indeed No Free Lunch over all functions, it follows that Binary is better over the class-2 functions with 2 minima. Again, we denote a function by $f = 1234$ for brevity. The functions on the right in the following list simply reverse the assignment of the values with respect to the functions on the left, so the number of minima do not change.

Functions	Min in F	Min in G	Min in B	Functions
1 2 3 4 ->	1	1	1	<- 4 3 2 1
1 2 4 3 ->	1	1	1	<- 3 4 2 1
1 3 2 4 ->	2	2	1	<- 4 2 3 1
1 3 4 2 ->	1	1	2	<- 2 4 3 1
1 4 2 3 ->	2	2	1	<- 3 2 4 1
1 4 3 2 ->	1	1	2	<- 2 3 4 1
2 3 1 4 ->	2	2	1	<- 4 1 3 2
2 1 3 4 ->	1	1	1	<- 4 3 1 2
2 1 4 3 ->	1	1	1	<- 3 4 1 2
2 4 1 3 ->	2	2	1	<- 3 1 4 2
3 1 2 4 ->	1	1	2	<- 4 2 1 3
3 2 1 4 ->	1	1	2	<- 4 1 2 3

```
Number of minima in Gray for class-1:  16
Number of minima in Binary for class-1:  24

Number of minima in Gray for class-2:  16
Number of minima in Binary for class-2: 8
```

What about functions with outputs that are ties? Our search algorithm will look at all Defining neighbors that are tied to see if there is an improving move from any neighbor with an equivalent output. If not, the equivalent neighbors are treated as a single optimum, since they share a single basin of attraction. Note that F already handles ties. Recall that by construction of F, ties are mapped to adjacent sets of integers in the range. First we look at how two strings can be tied. If the outputs ranked 3 and 4 are equivalent, it has no impact on the number of minima in the Defining neighborhood or the Gray or Binary Hamming neighborhoods and the number of optima induced is identical to the case where all 4 values in the range are unique. Over this subset of functions, Gray is again better than Binary for class-1 functions that have 1 minimum in the Defining neighborhood. If outputs ranked 2 and 3 are equal, then there is always a single optimum at 1 and Gray and Binary are equivalent for this subset. If outputs ranked 1 and 2 are equal, the result is the same as when all 4 values are unique; this is also true if $1 = 2$ and $3 = 4$, but $1 \neq 3$. There are only 3 other possibilities: $1 = 2 = 3$ and $1 = 4$; $1 \neq 2$ and $2 = 3 = 4$; and last $1 = 2 = 3 = 4$. In the last 3 cases, there is a single optimum and Gray and Binary are equivalent.

Thus, for the class-1 functions in F with a 2 bit representation, the corresponding set of Gray representations induces fewer *total* minima in the corresponding set of representations under a Hamming distance 1 neighborhood search compared to the set of Binary representations.

Generalizing to all L Bit Functions

THEOREM 1: *(The Gray-Compactness Theorem)* Let X be the number of local optima induced by a Hamming distance 1 neighborhood search operator over the bits in any Gray coded representation of a function in F. Let Y be the number of local optima induced under the Defining neighborhood topology of the same function in F: $X \leq Y$.

PROOF: By definition, a Gray coding ensures that all adjacent numbers in the Defining neighborhood differ by only one bit in Hamming space: this means that the adjacency of the Defining neighborhood is preserved in any Gray encoding. By induction, all paths in the Defining neighborhood topology are preserved in the Gray encoding. Since the original connectivity of the problem is preserved, no new optima can be created. The result automatically follows. **QED.**

The additional connectivity introduced by a Gray coding can create paths out of what are local minima in the Defining neighborhood topology, so that the Gray representation may have fewer optima.

Definitions:

Let FQ be the subset of functions in F with Q or fewer optima in their Defining neighborhood topology; $\overline{FQ}$ is FQ-complement, the subset of functions in F with more than Q optima.

Let RQ be the set of bit representations of functions in R with Q or fewer optima in their Hamming neighborhood topology; $\overline{RQ}$ is RQ-complement, the set of functions with more than Q optima in R.

Q is used to partition F and R as follows.

R: $\{.....RQ.....\}$ | $\{....\overline{RQ}....\}$

F: $\{.....FQ.....\}$ | $\{.....\overline{FQ}.....\}$

Q

Let $C(G, FQ)$ be the count of all optima in the Hamming neighborhood of Gray representations of functions in FQ. Let $C(G, \overline{FQ})$ be the count of all optima in the Gray representations of functions in $\overline{FQ}$.

Let $C(B, FQ)$ be the count of all optima in the Hamming neighborhood of Binary representations of functions in FQ. Let $C(B, \overline{FQ})$ be the count of all optima in the Binary representations of functions in $\overline{FQ}$.

LEMMA 1: *(NFL for bit representations)*

$$C(G, FQ) + C(G, \overline{FQ}) = C(B, FQ) + C(B, \overline{FQ})$$

PROOF: Every bit-encoded function in R is the Gray coding for some function in F; it is also the Binary encoding for some function in F. Thus, over all functions in F, the total number of local optima under each representation is the same. **QED.**

A useful corollary of Lemma 1 is:

$$C(G, FQ) < C(B, FQ) \iff C(G, \overline{FQ}) > C(B, \overline{FQ})$$

As will be seen, for specific values of Q, one can construct functions in FQ that have a Binary representation in $\overline{RQ}$. However, when Gray code is involved, the following Lemma holds.

LEMMA 2: Every function in FQ has a Gray representation, r_g in RQ. Every representation, $r_{g'}$ in $\overline{RQ}$ is the Gray representation of a a function in $\overline{FQ}$.

PROOF: Lemma 2 follows directly from the Gray-Compactness Theorem: every function in FQ has Q or fewer minima and must have a Gray representation in RQ; every representation in $\overline{RQ}$ is the Gray representation of some function that must have more than Q optima and hence must be in $\overline{FQ}$. **QED.**

THEOREM 2: For all discrete functions in F with bit representation of length L ($L > 1$), for $Q = 1$, the set of all Gray coded representation for functions in FQ induce fewer total minima than the set of all Binary coded representations for functions in FQ.

PROOF: When Q = 1, then $C(G, FQ) = |FQ|$, by the Gray Compactness theorem. In other words, each function in FQ that has 1 minimum in its Defining neighborhood topology can have only 1 minimum in its Gray representation. The number of minima in the Binary representation of each function in FQ can only be 1 or some value greater than 1. If there is any function with 1 minimum in its Defining neighborhood that has 2 minima in its Binary representation, then $C(B, FQ) > |FQ|$ and $C(B, FQ) > C(G, FQ)$. Consider the function, f_x where $f_x(0) = 4, f_x(1) = 2, f_x(2) = 1, f_x(3) = 3$, and otherwise, $f_x(i) = i + 1$. This function is in FQ since it has only 1 minimum in the Defining neighborhood, and so the Gray code of f_x must be in RQ, but the Binary code induces 2 minima and is in $\overline{RQ}$. This proves that for all L, $C(G, FQ) < C(B, FQ)$ when $Q = 1$. **QED.**

A corollary of the the theorem for $Q = 1$ is that $C(B, \overline{FQ}) < C(G, \overline{FQ})$ which means that Binary is better over $\overline{FQ}$ when $Q = 1$. This might seem to offer the "practical" advantage to Binary if most real world problems have more than 1 minimum, but there is additional evidence to suggest that Gray is better for a much wider set of values for Q than just $Q = 1$.

A worst case function in F occurs if every other number is a minimum. An analogous worst case function in R occurs if all strings with even numbers of bits are minima (or maxima) while all strings with odd numbers of bits are maxima (or minima). Thus define $MAX = 2^{L-1}$ as the maximum number of minima in the Defining and Hamming neighborhood topologies.

CONJECTURE 1: For all functions in F with bit representations of length L, when $Q = MAX - 1$ the set of all Gray coded representation for functions in FQ induce fewer total minima than the set of all Binary coded representations for functions in FQ.

Note that the conjecture is true if $C(G, \overline{FQ}) > C(B, \overline{FQ})$ when $Q = MAX - 1$. The conjecture is true for 3 and 4 bit functions. We now present Lemma's that provide evidence for Conjecture 1.

LEMMA 3. Let f_i be a function that has MAX optima in the Defining neighborhood. The Binary representation of f_i has at most 2^{L-2} optima.

PROOF: If f_i has MAX optima in the Defining neighborhood, then all even (or odd) integers are minima. The bit strings representing all even points end with a 0 and all odd points end with a 1 under a Binary representation. Thus all minima reside in an $L-1$ dimensional hypercube which can be constructed by taking all strings representing the 2^{L-1} minima and dropping the last 0 (or 1 if odd integers are minima). Only half the points in the resulting $L-1$ dimensional bit representation can still be minima. Thus, at most 2^{L-2} exist. **QED.**

LEMMA 4. Let f_i be a function that has MAX minima in the Defining neighborhood so that $f_i \in \overline{FQ}$ when $Q = MAX$. Let X be the maximum evaluation associated with any minimum. Let Y be minimal value associated with any maximum that separates minima. If $X < Y$, then any Gray code of f_i also has MAX minima.

PROOF: Assume f_i is an L-bit one dimensional

function. Under Gray coding, all adjacent numbers are 1 bit away in Hamming space. Therefore, all even numbers will be encoded using an even number of 1 bits in the string and all odd numbers with an odd number of 1 bits in the string, or vice versa. Thus, if $f_i \in \overline{FQ}$ when $Q = MAX - 1$ then any Gray coding of f_i isolates all of the minima in f_1 along any path that occurs in both the Defining neighborhoods or the Gray Hamming neighborhoods. The only way for minima to collapse is via the additional connectivity introduced in the higher dimensional bit representation space; but because every minimum has a lower evaluation than any maximum, every connecting path between minima must contain a maximum, and thus all minima are preserved. If f_i is an N-dimensional function, the same argument applies to every subspace of f_i so the result still holds. **QED.**

Lemma 3 and 4 show that while there are functions in $\overline{FQ}$ when $Q = MAX$ that induce $MAX = 2^{L-1}$ optima in the corresponding Gray representation, the same functions have at most 2^{L-2} optima in their Binary representations. Furthermore, all representations in $\overline{RQ}$ when $Q = MAX$ serve as the Gray representations of functions in $\overline{FQ}$ by the Gray compactness theorem. But by Lemma 4, all representations in $\overline{RQ}$ when $Q = MAX$ serve as the Binary representations of functions in FQ.

More Examples and Empirical Data

We previously enumerated all possible functions in F that can be mapped onto a 2 bit representation. This example is limited, however, since MAX in this case is 2 and $Q = 1 = MAX - 1$.

We can also enumerate all possible functions in F with unique outputs that can be mapped onto a 3 bit representation. We also divide F up into subsets that have *exactly* K minima in Defining space so as to preserve more information about the functions. Doing this we then obtain the following result.

K	No. of F with K minima	No. of Min in Gray	No. of Min in Binary
1	512	512	960
2	14,592	23,040	27,344
3	23,040	49,152	49,392
4	2,176	7,936	2,944

The construction of subsets of the form FQ (i.e., functions with Q or fewer minima) is unnecessary in as much as the trend over functions with exactly K minima is obvious. There are of course 8! or 40,320 functions and the number of minima for both Gray or Binary representations is 80,640. Note that Gray is better than Binary *for all* $Q = 1$ to $MAX - 1$. We speculate this is always true regardless of L.

For higher dimensional functions, enumeration quickly becomes impossible. We have randomly sampled 1,000,000 functions that can be mapped onto 5 bit encodings and 1,000,000 functions that can be mapped onto 6 bit encodings. In each case, Gray was better than Binary for all subsets of functions with *exactly* K minima in the Defining neighborhood as long as K was less than $\frac{2}{3}MAX$. However, empirically when K is between $\frac{2}{3}MAX$ and MAX Binary begins to dominate in terms of inducing fewer minima in the resulting bit representations than Gray.

Conclusions

We have shown that it is possible to partition the set of all possible functions so that for interesting subclasses a Gray representation induces fewer total minima than a Binary representations. The reverse is also true, but the evidence suggests that Gray is likely to be better for most common test problems, and perhaps most applications.

Acknowledgments

This work was supported by NSF grants IRI-9312748 and IRI-9503366. Soraya Rana was supported by a National Physical Science Consortium fellowship.

References

Caruana, R., and Schaffer, J. 1988. Representation and Hidden Bias: Gray vs. Binary Coding for Genetic Algorithms. In *Proc. of the 5th Int'l. Conf. on Machine Learning*. Morgan Kaufmann.

Davis, L. 1991. Bit-Climbing, Representational Bias, and Test Suite Design. In Booker, L., and Belew, R., eds., *Proc. of the 4th Int'l. Conf. on GAs*, 18–23. Morgan Kauffman.

Mathias, K. E., and Whitley, L. D. 1994a. Changing Representations During Search: A Comparative Study of Delta Coding. *Journal of Evolutionary Computation* 2(3):249–278.

Mathias, K. E., and Whitley, L. D. 1994b. Transforming the Search Space with Gray Coding. In Schaffer, J. D., ed., *IEEE Int'l. Conf. on Evolutionary Computation*, 513–518. IEEE Service Center.

Rana, S., and Whitley, D. 1997. Search, representation and counting optima. Technical report, Colorado State University.

Wolpert, D. H., and Macready, W. G. 1995. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute.