Search, Binary Representations and Counting Optima

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

Keywords: genetic algorithms, evolutionary computing, search, mathematical foundations

November 20, 1997

Abstract

Choosing a good representation is a vital component of solving any search problem. However, choosing a good representation for a problem is as difficult as choosing a good search algorithm for a problem. Wolpert and Macready's No Free Lunch theorem proves that no search algorithm is better than any other over all possible discrete functions. We elaborate on the No Free Lunch theorem by proving that there tend to be a small set of points that occur as local optima under almost all representations. Along with the analytical results, we provide some empirical evaluation of two representations commonly used in genetic algorithms: Binary Reflected Gray coding and standard Binary encoding.

Acknowledgements

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

1 Introduction

Wolpert and Macready's No Free Lunch theorem [5] proves that no search algorithm is better on average over the set of all possible functions than any other search algorithm. While this theorem is simple, it makes it clear (subject to certain assumptions) that there is no general purpose search algorithm that can solve all search problems better than any other algorithm.

We present a No Free Lunch result proving, for any function of fixed size N and any local search operator with a fixed neighborhood size k, that "good" local optima remain local optima under almost all representations of that function. We compute the expected number of local optima that will occur for any function and fixed neighborhood search operator under all possible representations. We also look at the practical limitations of this No Free Lunch result by illustrating how commonly used encodings measure up to the expected number of local optima for arbitrary functions. In practice, commonly used encodings for sample test problems reveal dramatically fewer local optima than expected for arbitrary representations. We also introduce a new technique for exploiting multiple binary representations of a search space.

2 No Free Lunch

There are few theoretical results that apply in a general fashion to search. Recently, there has been a good deal of interest in the "No Free Lunch" results as they apply to discrete function optimization and search. To understand these results, we first outline some of the simple assumptions behind this theorem. First, assume the optimization problem is discrete; this describes all combinatorial optimization problems–and really all optimization problems being solved on computers since computers have finite precision. Second, we ignore the fact that we can resample points in the space.

The "No Free Lunch" result can be stated as follows:

> The performance of all possible search algorithms is exactly the same when averaged over all possible functions.

Abstractly, we can represent an algorithm as an ordering, or permutation, over the points in the search space. We can also view all search algorithms as being deterministic; "stochastic" algorithms are, in practice, deterministic and can be modeled as a stochastic search operator coupled with a specific random seed. Thus, for any particular problem with a fixed search space of size N, an algorithm is just one of N! ordering of points in the search space. Furthermore, from this operational point of view, all possible representations of a search space is also the same set of N! orderings. Thus, there is an isomorphic relationship between all possible search algorithms over this collection of points, all possible functions that can be constructed from this set of points and even all possible representations over this set of points.

Note that if we fix the search algorithm, then these permutations represent all possible functions (or representations) over this set of points. On the other hand, if we fix the function (representation), then the permutations represent all possible search algorithms over this set of point. If we vary both search algorithms and functions, then every search algorithm is searching over the same set of permutations and the performance of all search algorithms are identical. Since this is true for any discrete set of points we might wish to pick, all search algorithms are the same over all possible discrete functions [3].

3 Representation and Local Optima

Given any discrete function of size N, one can compute the average number of optima that will occur over all local search operators with a fixed neighborhood of size k. We can also compute the number of points that remain local optima with a specific probability p.

Suppose we have N unique points in our search space and a search operator that explores k points before making its next move. (We will deal with points with duplicate evaluation in a later section.) A point is considered a local optima from a steepest ascent perspective if its evaluation is better than all of its k-neighbors. Further suppose that the N points in our search space each have unique values. We can sort those points to create a ranking, $R = r_1, r_2, ..., r_n$, in terms of their function values (where r_1 is the best point in the space and r_n is the worst point in the space). Using this ranking, we can compute the probability that a point ranked in the i-th position in r is a local optima under an arbitrary representation of the search space. This probability is given by the formula:

$$P(i) = \frac{\binom{N-i}{k}}{\binom{N-1}{k}} \qquad [1 \leq i \leq (N-k)] \tag{1}$$

Proof:

For any point in the search space, there are $\binom{N-1}{k}$ possible neighbors for that point. If the point is ranked in position r_1, then there are $\binom{N-1}{k}$ sets of neighbors that do not contain a point of higher evaluation than the point r_1. Therefore, the point ranked in position r_1 will always be a local optima under all representations of the function. In the general case, a point in position r_i has only $\binom{N-i}{k}$ sets of neighbors that do not contain a point of higher evaluation than the point r_i. Therefore the probability that the point in position r_i remains a local optima under an arbitrary representation is $\binom{N-i}{k}/\binom{N-1}{k}$. $\square$

As N increases, the probabilities quickly become difficult to compute. For implementation purposes, it is easier to represent the probabilities as a recurrence relation. The recurrence relation is given as:

$$P(i) = P(i-1)\frac{N-k-(i-1)}{N-(i-1)} \; [2 \leq i \leq (N-k)], \qquad P(1) = 1.0 \tag{2}$$

These probabilities enable us to count the expected number of local optima that should occur in any function of size N. The formula for computing the expected number of times a particular point will be a local optima is simply $N! \times P(i)$. Therefore the expected number of optima over the set of all representations is:

$$\mathcal{E}(N,k) = \sum_{i=1}^{N-k} P(i) \times N! \tag{3}$$

If we want to find the expected number of local optima for a single representation instance, we divide $\mathcal{E}(N,k)$ by $N!$, which yields:

$$\mu(N,k) = \sum_{i=1}^{N-k} P(i) \tag{4}$$

In addition to computing the expected number of local optima for an arbitrary function, we can compute the ranking index i for r_i for any specific probability. This information can be used to indicate how many points are likely to occur as optima in most representations. Suppose we are interested in knowing which r_i has a probability $P(i) \geq p$. Let $p = 1/X$ and assume $P(i) = p$. We now need to solve the following equation:

$$\frac{\binom{N-i}{k}}{\binom{N-1}{k}} = 1/X$$

It follows that:

$$X = \frac{\binom{N-1}{k}}{\binom{N-i}{k}} = \prod_{j=1}^{i-1} \frac{N-j}{N-k-j}$$

Note that we can compute bounds for this equation.

$$\left(\frac{N}{N-k}\right)^{(i-1)} \leq X \leq \left(\frac{N-(i-1)}{N-k-(i-1)}\right)^{(i-1)} \tag{5}$$

Let the lower bound estimate of i be represented by i_{LB}, which is computed as follows.

$$i_{LB} = \frac{ln(X)}{ln(\frac{N}{N-k})} - 1 \tag{6}$$

We next compute i_{UB} using the following:

$$i_{UB} = \left(\frac{N-(i_{LB}-1)}{N-k-(i_{LB}-1)}\right)^{(i_{LB}-1)} \tag{7}$$

Using these bounds, a search between the bounds can quickly produce the exact point i at which $P(i) \geq p$ even for very large search spaces.

3.1 Functions with Reoccurring Values

Most real-world functions are not restricted to producing unique function values. Therefore, we adapt the calculations in the previous section to the case where there are duplicate function evaluations.

Once again, we start with the ranking function R where all function evaluations are sorted (with duplicates). We then create a set of disjoint subsets, G. Each subset $g \in G$ contains indices into R such that for all $i \in g$, $r_i = y$ where y is a particular function evaluation. In other words, all elements of a subset index into R where all of those points have an equal evaluation. The groups that make up G can then be ordered based on evaluation, with the ranking of elements in g_k being better (less than for a minimization problem) than the evaluation of members of g_{k+i} for all positive integers i. The function G serves as a meta-ranking of the function R.

We must first tackle the problem of computing the probability that any set of points occurs as a local optima. The function P that was defined in the previous section is a one-to-one function with R. We can use P to define a new set of probabilities $\mathcal{P}$ that can be associated with the subsets in G. Using those new probabilities, we can compute the expected number of optima that will occur over the set of all functions (denoted $\mu(N, k)$). Finally, we can adapt the calculation that a point in a given position in R will occur as a local optima with probability of at least p.

Case 1: We define "locally optimal" points as points that are better than and not equal to any of their neighbors (i.e. we assume that points that occur on ridges and plateaus are not considered locally optimal).

The probability, $\mathcal{P}(k)$, for any point indexed by subset g_k to occur as a local optima is:

$$\mathcal{P}(k) = P(max(g_k)) \tag{8}$$

where the function $max(g)$ returns the maximum index contained in the set g. We can then define $\mu(N, k) = \sum_{k=1}^{m} |g_k| \times \mathcal{P}(k)$ where $m = |G|$.

Now we are interested in finding the index i' such that the point $r_{i'}$ occurs as a local optima with at least probability p. We already know how to compute the value i where $P(i) \geq p$ when there are no duplicates in R. We can use G and i to compute the new value i' that takes into account the duplicate function evaluations. First we locate g_k such that $i \in g_k$. Then we know the following information about $P(i)$: either $P(i) = \mathcal{P}(k)$ or $P(i) > \mathcal{P}(k)$.

$$i' = \begin{cases} i & if \;\; \mathcal{P}(k) = P(i) \\ max(g_{k-1}) & otherwise \end{cases} \tag{9}$$

Case 2: We now define locally optimal to mean that a point must be better than or equal to any of its k-neighbors. This is the case where a ridge or plateau is treated as locally optimal.

The probability $\mathcal{P}(k)$ of a subset of points indexed by the subset g_k occurring as local optima is:

$$\mathcal{P}(k) = P(min(g_k)) \tag{10}$$

where the function $min(g)$ returns the minimum index contained in the set g. The average number of local optima that one would expect to find in an arbitrary representation of the function is still $\mu(N, k) = \sum_{k=1}^{m} |g_k| \times \mathcal{P}(k)$ where $m = |G|$.

Once again we need to find the position i' in R such that the point $r_{i'}$ occurs as a local optima with at least probability p. Starting with the value i computed assuming no

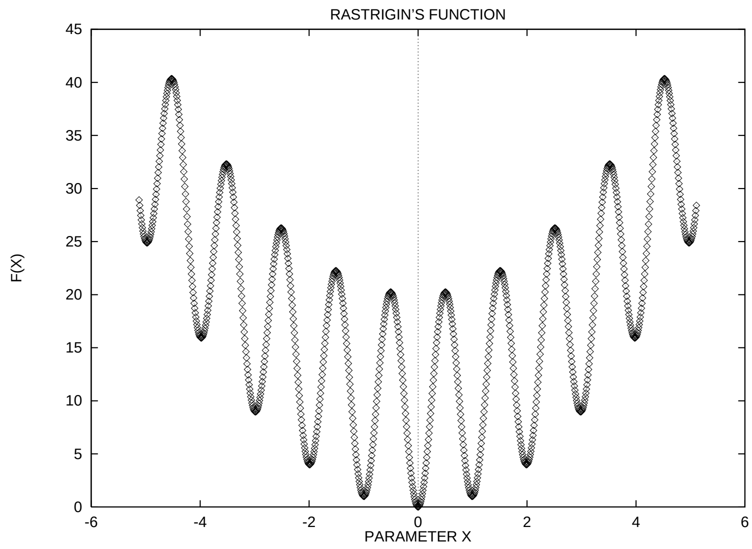


Figure 1: Rastrigin's function encoded using 1024 data points on a single dimension.

duplicates, we can generate the new index i'. First we locate g_k such that $i \in g_k$. Then we know that $P(i) \le \mathcal{P}(k)$ by definition. So the new index i' must be:

$$i' = min(g_k) \tag{11}$$

4 Analysis Of A Sample Function

The calculations in the previous section can be of use to genetic algorithm researchers in several ways. Since genetic algorithms have traditionally been used with binary encodings, these calculations can provide useful insight into the relative quality of different encoding techniques. Secondly, these calculations can also provide insight into the complexity of the underlying function landscape given some fixed binary encoding.

We looked at two possible binary encodings: Binary Reflected Gray Coding [1] and standard binary encoding. There are a large family of Gray codes (we speculate there are at least $L * 2^L$ such codes; the exact number is an open question).

Throughout this section we will analyze a specific multimodal function (Eq.12) known as Rastrigin's function [2] and illustrated in Figure 1.

$$f(x_i \mid_{i=1,N}) = (10N) + \left[\sum_{i=1}^{n} \left(x_i^2 - 10cos(2\pi x_i)\right)\right] \tag{12}$$
$$x_i \in [-5.12, 5.11]$$

This function is typically minimized over multiple dimensions with each parameter remaining independent of all others. If we look at Rastrigin's function in numeric space with a neighborhood size of two (looking only at a point on either side of our current point), then there are 11 local minima along any dimension. This also means that there are 11^N possible local minima in an N-dimensional version of the function.

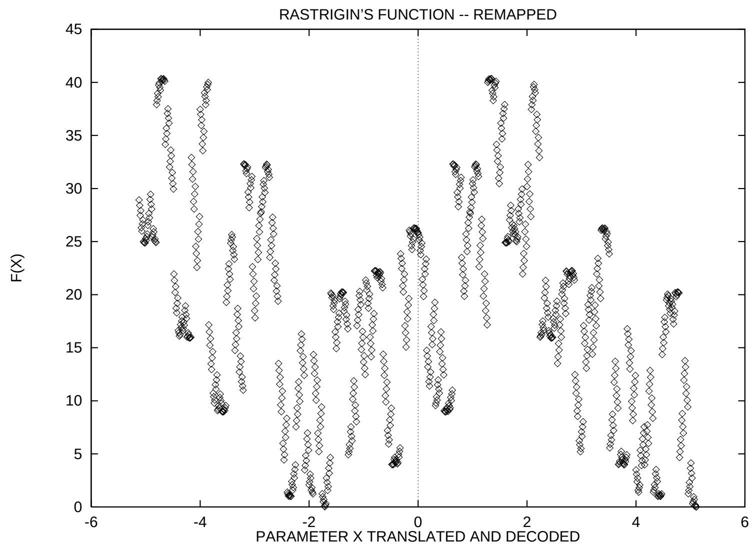


Figure 2: Function resulting from decoding a Binary representation of Rastrigin's function as a Gray code.

When we impose a binary encoding on the function, however, the number of locally optimal points that occur will change. The function is discretized using a 10-bit encoding. The probabilities of any point occurring as a local optima in a space of 1024 data points with a neighborhood size of 10 can be quickly computed using the recurrence relation. Since this function does contain duplicates, it is necessary to adjust the probabilities to account for the duplicates. Once this is done, the average number of local optima expected to occur in any random reordering of this function can be computed by summing the probabilities. That value is 93.5885. This value was empirically verified using 1000 random orderings of the function values yielding an average of 93.3450 locally optimal points.

Using Reflected Gray coding there are only 5 locally optimal points in Rastrigin's function. While there were 11 optima in numeric space, under standard Binary encoding there are 19 locally optimal points. Using our previous results, we note that given the expected number of local optima for an arbitrary binary encoding is nearly 93: the standard Binary representation is still much better than most other possible encodings. Taking these values to the context of an N-dimensional problem, it is clear that having 5^N is much more desirable than having 11^N or 19^N local optima.

Another interesting way to visualize the effects of the standard binary encoding on this problem is as follows. Let $\mathcal{R}_b$ be the representation obtained by converting from integers representing the numeric input space for Rastrigin's function to bits using a standard Binary encoding. Now assume we have a function $G(x)$ that when encoded in Binary and Grayed (using Binary Reflected Gray coding) also results in $\mathcal{R}_b$. A Gray encoding $\mathcal{R}_b$ preserves the connectivity of the numeric function $G(x)$ so that all points adjacent in numeric space are adjacent in Hamming space [4]. The resulting function $G(x)$ is shown in Figure 2. This image illustrates the additional roughness introduced into the topology of the function when standard Binary encoding is used.

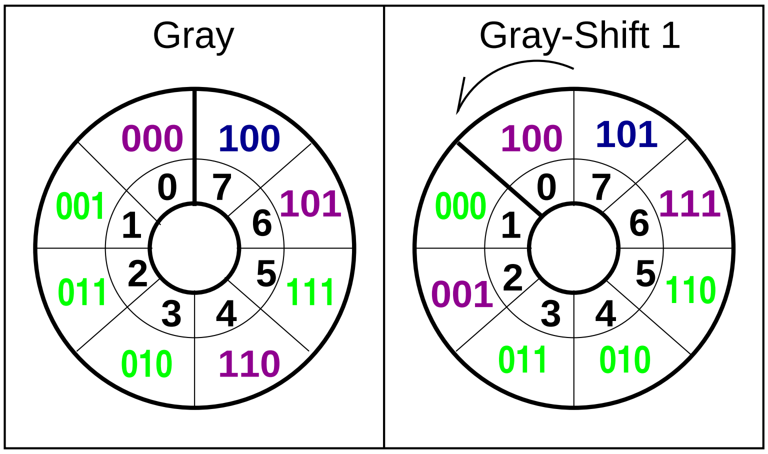


Figure 3: Example of shifting using a Gray encoding.

4.1 Changing Representation by Shifting

It is well understood that arbitrary recodings of the problem will probably not reduce the number of local optima in a function and in the case of a well-behaved function like Rastrigin's function, it is likely to induce more local optima into the function. However, it is possible to generate small modifications to the representation of a function in a controlled fashion so that the change to the landscape is small. This type of change could be effectively used when running many search algorithms in parallel using different representations. To this end, we introduce **shifting**.

To understand shifting consider figure 3 which illustrates shifting Gray coded 3-bit strings. Consider the outer circle as a dial that can be turned counterclockwise. The inner circle contains the number to which each string is mapped. Shifting is done by turning the dial to create a new mapping. The right half of Figure 3 shows the Gray coded 3-bit strings with a single shift applied.

A shift in any Gray representation just changes the representation to a different Gray code, since the adjacency of the numbers is preserved. Applying a shift to an encoding does not randomly shuffle the points underlying the bit encoding. However, it alters the structure of the neighborhood of a given point. Take the value 7 in Figure 3, for example. The neighbors of the value 7 have changed with a simple shift of one. By changing the neighborhood structure of a given encoding, one should expect changes to the number of local optima in the new function mapping.

The next section illustrates the effects of shifting under binary and Gray encodings for 3 multimodal test problems. We look at the previously defined Rastrigin's function as well as functions developed by Schwefel (Eq.13) and Griewangk (Eq.14) [2].

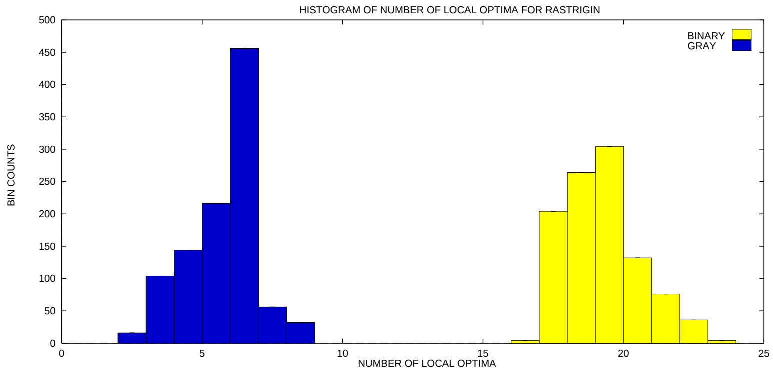
$$f(x_i\mid_{i=1,N}) = \sum_{i=1}^{N} -x_i sin(\sqrt{\mid x_i \mid}) \tag{13}$$

$$f(x_i\mid_{i=1,N}) = 1 + \sum_{i=1}^{N} \frac{x_i^2}{4000} - \prod_{i=1}^{10}(cos(x_i/\sqrt{i})) \tag{14}$$

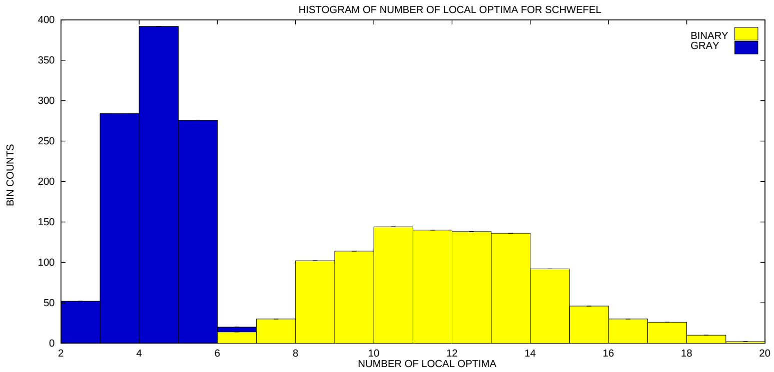
$$x_i \in [-512, 511]$$

All three functions were analyzed for a single parameter represented using 10 bits.

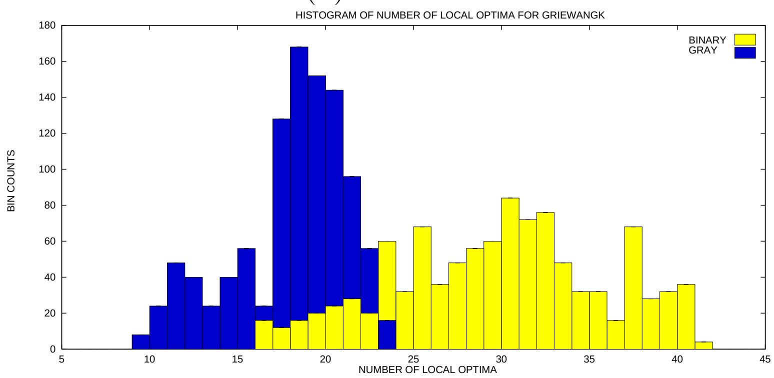
4.2 Effects of Shifting



(a) Rastrigin



(b) Schwefel



(c) Griewangk

Figure 4: Histograms of the number of local optima found in Binary and Gray spaces

In order to examine what kinds of changes can be induced by shifting the binary encoding of a function, we applied the 1024 possible shifts (for a 10-bit encoding) to all three test functions. Figure 4 shows the histograms of number of optima (assuming minimization) generated over 1024 shifts of both Binary and Gray encodings where the black bars represent the Gray encodings. Table 1 provides some statistics for the histograms in Figure 4. For all three test functions, the shifting generates fairly tight distributions which indicates that the absolute number of local optima is not radically changed. In all cases, the number of local optima for all shifts is still far from approaching the expected number of local optima under arbitrary representations. It is clear from the histograms that shifting the space does vary the number of local optima in a controlled manner. However, it says nothing about which local optima occur.

	Numeric	Binary		Gray	
Function	#Opt	μ	σ	μ	σ
Rastrigin	11	19.69	1.21	5.26	1.25
Schwefel	8	11.38	2.57	2.93	0.91
Greiwangk	163	29.73	6.21	17.58	3.23

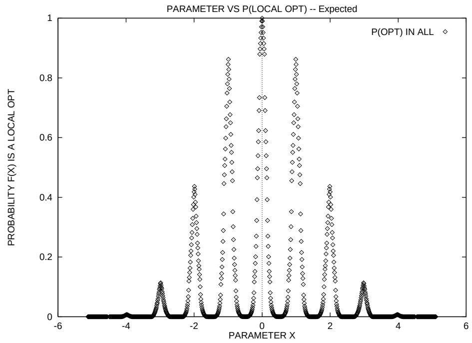
Table 1: Mean (μ) and standard deviations (σ) for number of optima found for 1024 shifts of three test functions.

Rastrigin's function contains a squared term that causes duplicate parameter evaluations. This means that a locally optimal value can be induced by two different parameter values with the exception of the parameter value zero. Rather than looking only at which locally optimal values occur under different representations, it is more informative to examine which locally optimal parameters occur under different representations. Figure 5 contains three graphs of probabilities that any parameter will be locally optimal. The first graph (a) presents the probability that any parameter will be a local optima under all possible representations. The graphs (b) and (c) represent the probability that any parameter will be an optima under any shift of Binary or Gray coding, respectively. For graphs (b) and (c), a vertical line was drawn to indicate how often each parameter occurred as a local optima. In the Binary case, notice that 5 parameters that will occur with probability 1 as local optima. However, around each of those 5 parameters is a blackened area that is due to a large number of parameters that occasionally occur as local optima. The actual number of parameters that occur as local optima in Binary space over all shifts is 346 (more than one third of the space). In Gray space, there is only 1 out of 9 local optima that will always occur under all shifts. In both cases, the implication is that if multiple shifts were effectively used together, the search space could become easier.

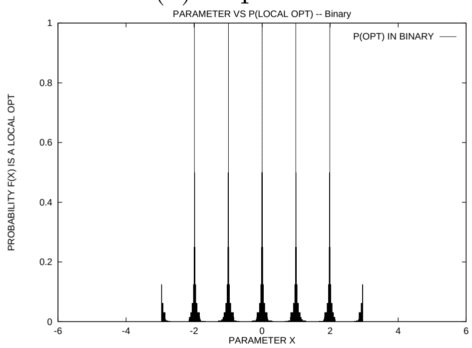
5 Conclusion

This paper has presented several new ideas that can be used to help genetic algorithm researchers understand how encoding can affect search performance. While it has already been shown that all functions of a particular discretization have the same number of local optima [3], we can compute how many local optima will occur on average for that set of functions. We also introduced methods for determining which points are likely to be optima under all representations.

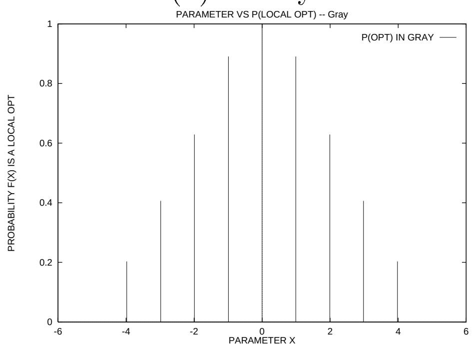
In practice, researchers do not deal with random functions. There is usually some continuity or gradient information in the function landscape. In other words, optima are often surrounded by relatively good points. When genetic algorithm researchers use a binary encoding, the complexity of the remapped function can be quite different than its Euclidean



(a) Expected



(b) Binary



(c) Gray

Figure 5: Probability that a parameter occurs as a local optima under an all shifts.

counterpart. We have illustrated how the landscape changes in a simple problem using two commonly used binary encodings, Standard Binary and Reflected Gray encodings.

We also introduced a new and simple mechanism for altering a problem encoding. Shifting does not drastically alter the number of local optima under an encoding but alter which points occur as local optima. We believe that using multiple shifts simultaneously in parallel may offer performance benefits.

References

[1] 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.

[2] H. Mühlenbein. Evolution in time and space: The parallel genetic algorithm. In G. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 316–337. Morgan Kaufmann, 1991.

[3] Nicholas J. Radcliffe and Patrick D. Surry. Fundamental limitations on search algorithms: Evolutionary computing in perspective. In Jan van Leeuwen, editor, *Lecture Notes in Computer Science*, volume 1000. Springer-Verlag, 1995.

[4] Darrell Whitley, Keith Mathias, Soraya Rana, and John Dzubera. Evaluating evolutionary algorithms. *Artificial Intelligence Journal*, 85, August 1996.

[5] David H. Wolpert and William G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute, July 1995.