Gray, Binary and Real Valued Encodings: Quad Search and Locality Proofs

Darrell Whitley[1] and Jonathan Rowe[2]

[1]Computer Science, Colorado State University, Fort Collins, CO 80523
[2]Computer Science, University of Birmingham, Birmingham B15 2TT, UK

Abstract. This paper looks at proofs concerning Gray codes and the locality of local search; it is shown that in some cases such proofs can be generalized to Binary and real valued encodings, while in other cases such proofs do not hold for Binary encodings. The paper also looks at a modified form of Quad Search that is proven to converge in under 2L evaluations to a global optimum on unimodal 1-D bijective functions and to a local optimum on multimodal 1-D bijective functions. Quad Search directly exploits the locality of the Gray code representation.

1 Introduction

There have been numerous debates about the relative advantages of search algorithms that use Gray code representations compared to Binary and real-valued representations [CS88] [Dav91]. Reflected Gray codes are characterized by a symmetric reflection property, such that a point in one half of a 1-D space folds onto its neighbor in the other half of the search space. This property allow the construction of relatively simple recursive proofs related to the time required to convergence to a local optimum under local search, as well as the convergence of a specialized search strategy called Quad Search [WGW03]. The reflection properties of Gray code are also the foundation for another (new) proof that counts the number of neighbors which in expectation are in the same basin of attraction (or quasi-basin) as a given reference point; the proof assumes one is provided with information about the size of the basin of attraction relative to the size of the search space.

A modified form of Quad Search algorithm is introduced that allows it to provably converge to a local optimum when run on 1-D multimodal functions. An example shows that the original form of Quad Search does not always converge to a local optimum when searching 1-D multimodal functions.

But do these proofs imply that similar proofs do not hold for Binary representations? And what about real valued representations? When possible we constructs similar convergence proofs for Binary and real-valued representations. This paper also compares the assumptions and special purpose operators that are needed for the various convergence proofs.

New proofs are also presented concerning the likelihood of neighbors falling into the "quasi-basin" around the point currently visited under local search. The significance of these results relate to the No Free Lunch theorem [WM95] and the Christensen and Oppacher "Submedian Seeker" algorithm [C01].

2 Gray and Binary Encoding and Quadrants

Gray codes have at least two important properties. First, for any 1-D function (and in each 1-D slice of any multidimensional problem) adjacent neighbors in the real space are also adjacent neighbors in the Gray code hypercube graph [RW97]. Second, the "standard Binary reflected Gray code" folds the search space in each dimension, creating a symmetric reflected neighbor structure [BER76] [RWB04]. For any reference point i, there is exactly one neighbor in the opposite half of the search space. For example, in a 1-D search space of points indexed from 0 to $2^L - 1$, the points i and $(2^L - 1 - i)$ are neighbors. In effect, these neighbors are found by folding or reflecting the 1-D search space about the mid-point between 2^{L-1} and $2^{L-1} + 1$.

There are also reflections between each quartile of the search space. Starting at point i in the first quartile of the search space we can define a set of four integer values where each integer falls in a different quartile, or "quadrant" of the search space. The four points are as follows:

$$\begin{aligned} i &= i \\ (2^{L-1} - 1) - i &= i \oplus (2^{L-1} - 1) \\ (2^L - 1) - i) &= i \oplus (2^L - 1) \\ (2^{L-1} + i) &= i \oplus (2^L - 1) \oplus (2^{L-1} - 1) \end{aligned} \tag{1}$$

where $\oplus$ denotes exclusive-or on the Binary encoding of the integer. Both $2^L - 1$ and $2^{L-1} - 1$ are Hamming distance 1 from the string of all zeros. Therefore, moving from point i to one of the other 3 points involves flipping the first bit, the second bit, or both. Because exclusive-or just applies a bit-flip mask, this indexing scheme works regardless of the quadrant in which the point i actually appears.

2.1 Quad Search

Consider a unimodal function encoded using an L bit Gray code. Any given point has two neighbors in quadrants outside of its own quadrant; they are the points generated by flipping the two high order bits. The highest order unfixed bit is called the *major* bit. The second highest order unfixed bit is called the *minor* bit. Flipping any other bit must generate a neighbor inside the current quadrant.

Let BB denote the first two bits that map to an arbitrary quadrant. In effect, the pattern $BB{*}{*}\ldots{*}$, defines a hyperplane that corresponds to a (hypothetically) continuous quarter, or *quadrant* of the search space. Let N_{BB} refer to a point in quadrant BB. We can flip the major bit to generate the point at $N_{\overline{B}B}$ and the minor bit to generate the point at $N_{B\overline{B}}$. We flip both bits to reach the point at $N_{\overline{BB}}$. Each point is in a different quadrant, each with its respective index.

In Figure 1 the point located at X is a neighbor of point N_{BB} that is located in quadrant BB. X can be left or right of N_{BB}. One of the 4 quadrant neighbors is the current point from which the search is being carried out.

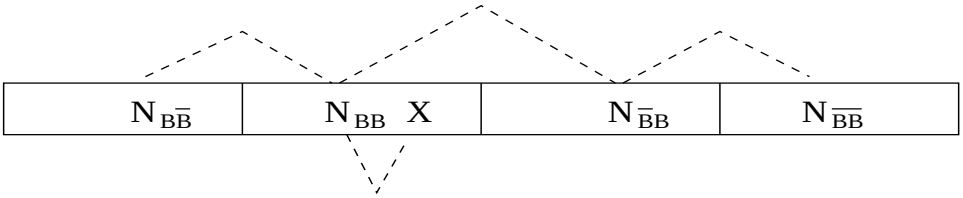

Fig. 1. The neighbor structure across and within quadrants.

```
\*************************************************\
\*  BEGIN ONE-DIMENSIONAL QUAD SEARCH ALGORITHM *\
\*************************************************\
```

Step 1: Randomly select point N_{BB}. Evalute point N_{BB} and
its reflected neighbors $N_{\overline{B}B}$, $N_{B\overline{B}}$ and $N_{\overline{BB}}$.

Step 2: Without loss of generality, relabel the points so that N_{BB} is the best of these four points.

Step 3: Flip the third unfixed bit in N_{BB} to create point X, and evaluate.
If $f(X) < f(N_{BB})$ set CASE = 1; else CASE = 2.
/* Samples a reflected neighbor in quadrant BB */
/* CASE refers to Case 1 and Case 2 of the proof */

Setp 4: Let $\mathbf{b}$ represent the third bit in the best-so-far solution.
If $\mathbf{b} = 0$ fix the minor (second) bit.
If $\mathbf{b} = 1$ fix the major (first) bit.
/* The current best must be in quadrant BB. The third bit determines if */
/* the local optima is bounded by the leftmost or rightmost quadrant of BB. */

Step 5: Relabel the points so that the new best-so-far is denoted N_{BB} and its reflected
neighbors $N_{\overline{B}B}$, $N_{B\overline{B}}$ and $N_{\overline{BB}}$. Three of these points have been evaluated.
/* This follows from the Quad Search convergence proof. */

Step 6: If CASE = 1, the unevaluated point is at $N_{\overline{B}B}$. Evaluate $N_{\overline{B}B}$.
/* If CASE = 2, the additional point is not evaluated. */

Setp 7: If there are more than 2 bits unfixed bits, goto Step 3. Otherwise, STOP.

```
\**********************\
\*  END QUAD SEARCH    *\
\**********************\
```

Fig. 2. The Quad Search algorithm. The proof uses an exclusive-or transform. This makes the proof more direct, but is not needed for the algorithm. However, the algorithm in the untransformed space behaves differently in terms of when the major and minor bits are fixed. Since only the first 3 bits are involved in both algorithm and proof, these can be shown to behave the same by enumeration of the 16 possible conditions (8 bit patterns, and CASE = 1 or CASE = 2).

A previously published paper introduces the original *Quad Search* algorithm and proves its convergence time [WGW03]. The new algorithm (see Figure 2) and the proof presented here have been modified to evaluate fewer points and to allow convergence proofs for multimodal 1-D functions. We express the problem in terms of minimization.

Theorem 1: *Quad Search converges to the global minimum on bijective unimodal 1-D functions and to a local minimum on multimodal 1-D functions after at most* $2L - 1$ *evaluations from an arbitrary starting point.*

Proof: The proof is recursive in nature. Assume the search space contains at least 8 points, so that there are at least 2 points in each quadrant. After initializing the search, we show the search space can be cut in half after at most 2 new evaluations.

Sample a point N_{BB} and its reflected neighbors $N_{\overline{B}B}, N_{B\overline{B}}$ and $N_{\overline{BB}}$.

Without loss of generalization, let N_{BB} be the current best solution. Let Z represent a string of zeros of arbitrary length. There exists a bit mask, m, which relabels the points $\left\{N_{BB}, N_{\overline{B}B}, N_{B\overline{B}}, N_{\overline{BB}}\right\}$ using exclusive-or so that the minimal neighbor is represented by the string $000Z$. The neighborhood structure is unchanged under exclusive-or. Transforming all the points yields:

$$N_{BB} \oplus m = 000Z, \;\; N_{\overline{B}B} \oplus m = 100Z, \;\; N_{B\overline{B}} \oplus m = 010Z, \;\; N_{\overline{BB}} \oplus m = 110Z$$

where $\oplus$ denotes exclusive-or. We sample one additional point at $001Z$ in quadrant 00.

We first deal with unimodal functions: note the minimum cannot be in quadrant 11.

Case 1. If $f(000Z) > f(001Z) < f(010Z)$, then the global optimum must reside in quadrant 00 or 01. The major bit is fixed to 0 and the search space is reduced. The points $000Z, 001Z, 010Z$, become the points $00Z, 01Z, 10Z$ in the reduced space. Only 2 additional points are needed to continue the search. First we evaluate $11Z$. Then the space is remapped again so that the minimum of these 4 points is at 00Z. Then we evaluate $001(Z-1)$.

Case 2. If $f(100Z) > f(000Z) < f(001Z)$, then the global optimum must reside in quadrant 00 or 10. The minor bit is fixed to 0 and the search space is reduced. The points $000Z, 001Z, 100Z$, become the points $00Z, 01Z, 10Z$ in the reduced space. We do not evaluate $11Z$ since we know $f(100Z) > f(000Z) < f(001Z)$ and the minimum of these 3 points is already at 00Z. We evaluate only 1 additional point at $001(Z-1)$.

After the first 5 evaluations, the search space contains $2^{(L-1)}$ points. At each iteration, the search space is cut in half after at most 2 new evaluations. After (L-3) iterations the search space is reduced to at most 4 points, since $2^{(L-1)}/2^{(L-3)} = 4$. However, at this point, we have already evaluated $00, 01$ and 10 (remapped so that 00 is the best solution so far); therefore we do not evaluate 11. The total number of evaluations is at most $5 + 2(L-3) = 2L - 1$. This proves convergence for the unimodal case.

To prove convergence for the multimodal case, it suffices to show that as sections of the search space are eliminated, the remaining space is a continuous interval; if the remaining space is a continuous interval of the original function (where the original 1-D function is defined to wrap around), then cases 1 and 2 above show that the remaining space contains a local optimum of the original function.

The following are specific Subcases of Case 1 and Case 2 above when the best solution so far is an *arbitrary* quadrant. In this case, we don't care where the best solution so far is located, only which quadrants are actually eliminated.

Subcase 1a. Fixing the major bit to 1 eliminates quadrants 00 and 01.
Subcase 1b. Fixing the major bit to 0 eliminates quadrants 10 and 11.
Subcase 2a. Fixing the minor bit to 1 eliminates quadrants 00 and 10.
Subcase 2b. Fixing the minor bit to 0 eliminates quadrants 01 and 11.

Note that under Gray code, the quadrants are (in order) 00, 01, 11, 10. Fixing the major bit (Subcases 1a and 1b) eliminated either the left half of the right half of the space, so that the reduced search space is a continuous interval of the search space at the previous iteration. Fixing the minor bit to 1 (Subcases 2a) eliminates the first and last quadrant, so that again the reduced search space is a continuous interval of the search space at the previous iteration.

Only Subcase 2b "fragments" the search space into two parts, except quadrants 00 (the first) and 10 (the last) are adjacent in the wrapped Gray encoding. Note that in Case 2 above, when a minor bit is set, we automatically eliminate quadrant 11 at the next iteration. Therefore, after Subcase 2b occurs only two possible alternatives exist: Subcase 1b or Subcase 2b must occur.

If Subcase 1b holds, the last half of the search space is eliminated, and if the space was previously fragmented into two parts, one of those parts is discarded, making the search space a continuous interval again.

If Subcase 2b holds again, the space is not remapped (the best known is already in quadrant 00). Therefore, eliminating quadrants 01 and 11 again means that the search space remains in two parts but still adjacent in the wrapped sense.

This means that the search space is never fragmented into more than two parts, and those two parts must always be adjacent in the wrapped sense. Cases 1 and 2 both insure that a local optimum is contained within the interval still to be searched. □

Obviously, many functions are not bijections. If ties in evaluation are not too common, the algorithm should still display linear time convergence. In general, convergence times that are less than exponential cannot be guaranteed if the space is largely flat.

As bits are fixed, the high order bits used in the computation refer to the two highest order bits *which remain unfixed*. The major bit is always the leftmost bit that has not been fixed. The minor bit is always the second bit from the left that has not been fixed.

2.2 Observations on the Search Behavior of Quad Search

The original Quad Search algorithm provides a form of local search that hybridizes extremely well with genetic algorithms [WGW03]. However, the original version of Quad Search has different behavior in Case 2 which can be summarized as follows.

Previous Case 2. When $f(100Z) > f(000Z) < f(001Z)$, the minor bit is fixed to 0. The points $000Z, 001Z, 100Z$, becomes the points $00Z, 01Z, 10Z$ in the reduced space. Evaluate $11Z$ and $001(Z-1)$ and continue.

Note that string $11Z$ is evaluated in this version, even though the local optimum is known to be in the interval between $f(10Z) > f(00Z) < f(01Z)$. If $f(11Z) < f(00Z)$ then the search will move to string $11Z$, but there is no longer an interval known to include a local optimum.

This can cause the original Quad Search to converge to a point that is not locally optimal on a multimodal function. The following example shows one can be as many as $2^{L-2} - 2$ positions away from a local optimum.

Assume we have N points numbered from $0...2^L - 1$. The quadrants correspond to the following intervals.

$$0 \ldots 2^{L-2} - 1; \;\; 2^{L-2} \ldots 2^{L-1} - 1; \;\; 2^{L-1} \ldots 3(2^{L-2}) - 1; \;\; 3(2^{L-2}) \ldots 2^{L-1} - 1$$

Assume we test reflected points located at $0, 2^L - 1, 2^{L-1}, 2^{L-1} - 1$ and find

$$f(2^L - 1) > f(0) < f(2^{L-1} - 1)$$

We next sample at location $2^{L-2} - 1$ and find $f(2^{L-1} - 1) > f(0) < f(2^{L-2} - 1)$. We eliminate half of the space from point 2^{L-2} to $3(2^{L-2}) - 1$.

The original Quad Search next sampled at reflection point $3(2^{L-2})$. Note this is not required, since a local optimum is isolated in the interval $2^{L-1} - 1 \ldots 2^{L-2} - 1$.

Assume $f(3(2^{L-2})) < f(0)$ and the search moves to this new best-so-far. It is possible for the function to be monotonically decreasing from point $3(2^{L-2})$ to $2^{L-1} + 1$; thus, there can man be as many as $2^{L-2} - 1$ improving moves leading to a local optimum located in an eliminated quadrant.

Of course, the original version of Quad Search was only designed to converge on unimodal functions; the modified Quad Search converges to a local optimum on 1-D multimodal bijective functions by only evaluating points in the interval known to contain a local optimum. And the modification to the algorithm is minor.

Another important observation involves running Quad Search on multidimensional bijective functions. Quad Search will *iteratively* converge to a local optimum in each dimension, but the point where Quad Search terminates after exploring each dimension once is only guaranteed to be locally optimal in the last dimension. In this regard, Quad Search is like a line search algorithm. For example, on a 2-dimensional function, dimension 2 is initially fixed to some arbitrary value, while Quad Search locates a local optimum in dimension 1; next the algorithm is run on dimension 2 while dimension 1 is fixed to the point which was found to be locally optimal. Of course, after a local optimum is found in dimension 2, the current best solutions may no longer be locally optimal in dimension 1. However, one can reapply Quad Search (as with line search) until a local optimum has been reached.

It is also important to note that if Quad Search is run on a separable bijective multimodal function, it will converge to a local optimum. And if the separable function is unimodal (for example, as is the sphere function [B̈96]), then Quad Search will find the global optimum in at most $2L - 1$ evaluations.

2.3 Simple Binary Search for Real Valued Encodings

A similar search can be done using real valued representation that also cuts the search space in half after 2 evaluations. We again assume a 1-D bijective function.

Evaluate points at 0, N/4, N/2, 3N/4, N-1 (rounding if necessary). Note this explicitly divides the space into 4 regions. After the minimum is identified, an optimum must lie in one of two quadrants. For example, let F be the evaluation function. If

$$F(0) > F(N/4) < F(N/2) < F(3N/4) < F(N-1)$$

then the optimum must be between $F(0)$ and $F(N/2)$. This is both simple and obvious. The process eliminates at least half (+/- 1 with rounding) of the search space from further consideration, and is guaranteed to converge to an optimum in at most $2(\log_2(N) + 1)$ evaluations. This search is very similar to Quad Search, except the resulting search does not search from an arbitrary point.

2.4 Binary Encodings and Quadrants

One of the reasons that Quad Search works using a Gray encoding is because the quadrants themselves have bit prefixes that are also adjacent and wrap. Fixing the first bit to 0 eliminates quadrants 3 and 4. Fixing the first bit to 1 eliminates quadrants 1 and 2. Fixing the second bit to 0 eliminates quadrants 2 and 3. Fixing the second bit to 1 eliminates quadrants 1 and 4. Thus, fixing one of the first two bits will eliminate two adjacent quadrants (where 1 and 4 are adjacent in the wrapped Gray encoding).

This does not happen with Binary neighbors. Consider a unimodal function with an optimum located at a Hamming cliff. One cannot fix bits and limit the search to quadrants 2 and 3 because the points in quadrant 2 start with the prefix 01 under the Binary encoding and the points in quadrant 3 start with the prefix 10. The first bit is either 0 or 1, or the second bit is either 0 or 1. Nothing is resolved. Because of the Hamming cliff, local search under a Binary encoding is not guaranteed to converge to a local optimum. This is already clear from results reported by Whitley et al. [Whi99] with regard to No Free Lunch: a Binary encoding can induce false local optima, even in a 1-D unimodal bijective function.

Of course one can use special operators to force a Binary search. Given a unimodal 1-D bijective function, one can evaluate the two points that make up the dominant Hamming cliff (complementary bit strings located at adjacent points), and determine the gradient at the Hamming Cliff and therefore determine in which half of the space the global optimum in located. The more important point, however, is that local neighborhood search in the standard Binary encoding neighborhoods does not have the same nice convergence properties as a reflected Gray code neighborhood.

3 Quasi-basins, Encodings and Locality

We formally define a **quasi-basin** for a 1-D function, f, with respect to a threshold value, V, where V is a codomain value of f: a quasi-basin is a set of contiguous points in f that are below value V. This in section, we present new proofs that outline sufficient

conditions to ensure that the majority of Hamming distance 1 neighbors under Gray and Binary encodings are either in the same basin of attraction, or in the same quasi-basin.

Consider some reference point R in the search space. Whitley, Rowe and Bush [WRB04] show that given a 1-D function, the number of neighbors that are $<= D$ points away from the reference point under Gray code is $\lfloor \log D \rfloor$.

Expressed another way, consider a quasi-basin of size D and a search space of size N where the quasi-basin spans $1/Q$ of the search space (i.e., $D = \frac{1}{Q}N$): under a Gray encoding at most $\lceil \log(Q) \rceil$ bits encode for points that are more than a distance of D points away from R. Note that an increase in precision also increases the size of the search space, so that the quantity N/Q becomes larger and thus $\log(N/Q)$ increases. However Q and $\log(Q)$ remain constant. Thus, at higher precision, the number of neighbors within a distance of N/Q points increases.

Whitley, Rowe and Bush [WRB04] also present the following result without a proof.

Given a quasi-basin that spans $1/Q$ of a 1-D function of size $N = 2^L$ and a reference point R inside the quasi-basin, the expected number of neighbors of R that fall inside the quasi-basin under a reflected Gray code is greater than $\lfloor \log(N/Q) \rfloor - 1$.

In the current paper we provide a proof, but also prove a more general result that allows us to make a similar statement about standard Binary encodings. We first present the proof for the reflected Gray code case, then show that similar bounds hold for the standard Binary encoding.

The significance of this result is that we can outline conditions that would allow a steepest ascent local search bit climber to spend the majority of its time sampling points that are contained in the same quasi-basin and therefore below threshold. This makes it possible to outline sufficient conditions such that steepest ascent local search is provably better than random search. We conjecture that these same conditions allow next-ascent bit climbing as well as Quad Search to out perform random search.

To prove these results, a number of supporting concepts and lemmas are needed.

4 Locality and Neighborhoods

We first define Gray and Binary encoding recursively in order to define other concepts. The Gray encoding is the Standard Binary Reflected Gray code. For strings of length 1, Gray and Binary encodings are the same: the strings 0 and 1 represent integers 0 and 1. We then recursively define either a Binary or Gray code as follows.

Let B_i denotes a Gray encoded string of length L representing integer i, where $0 \leq i \leq 2^L - 1$. Strings of length $L + 1$ are constructed by concatenation and have the form $0B_i$ or $1B_i$ and are defined as follows:

$$0B_0, 0B_1, 0B_2, ...0B_{2^L-1}, 1B_{2^L-1}, ..., 1B_2, 1B_1, 1B_0$$

Let B_i denotes a Binary encoded string of length L representing integer i, where $0 \leq i \leq 2^L - 1$. Strings of length $L + 1$ are constructed by concatenation and have the form $0B_i$ or $1B_i$ and are defined as follows:

$$0B_0, 0B_1, 0B_2, ...0B_{2^L-1}, 1B_0, 1B_1, 1B_2, ..., 1B_{2^L-1}$$

While these definitions are well known and obvious, we note that the Gray code folds the search space around a $reflection$ located between $0B_{2^L-1}, 1B_{2^L-1}$. In the Binary case, there is an analogous mid-point $transition$ between $0B_{2^L-1}, 1B_0$ in the recursive construction. We will refer to both of these as *transition points*; these points are important when documenting the locality of Binary and Gray encodings.

The placement of any transition point automatically implies the location of other transition points under the recursive definitions of both Binary and Gray code. We can define an arbitrary *key-transition* around which neighborhoods can be defined.

We will define **core neighborhoods** as 2^k adjacent points with k connections that are fully contained within the core neighborhood. Core neighborhoods need not have the same connectivity, but all members have the same *number* of core neighbors. Binary and Gray codes have the same "core neighborhoods" for any set of 2^k points that are within the quasi-basin and adjacent to a transition point,

We will assume that a *key-transition point* could occur at any position within the quasi-basin. We will count over all possible placements of transition points with the quasi-basin, then characterize what neighborhood structures occur for each possible transition point placement.

For example, let | denote the placement of a *key-transition* point in a quasi-basin made up of 7 points. The following represents the number of core neighbors under both Gray and Binary and the key transition denoted by | shifted into each possible position.

```
0 1 1 2 2 2 2 |
  1 1 2 2 2 2 | 0
    1 2 2 2 2 | 1 1
      2 2 2 2 | 1 1 0
        0 1 1 | 2 2 2 2
          1 1 | 2 2 2 2 0
            0 | 2 2 2 2 1 1
              | 2 2 2 2 1 1 0
```

All of this is obvious given the recursive definitions of Binary and Gray encodings. However we are interested in how Binary and Gray encoding differ in the construction of the non-core neighborhood connections.

4.1 The Matrix $\mathbf{M}^x$

We define a lower triangle matrix $\mathbf{M}^x$ using a recursive definition such that $\mathbf{M}^1 = [1]$. For $x > 1$ the lower triangle matrix $\mathbf{M}^x$ can be decomposed into a 2^{x-1} by 2^{x-1} square matrix whose elements are all the integer x, plus 2 identical copies of lower triangle matrix $\mathbf{M}^{x-1}$. The square matrix occupies the first 2^{x-1} columns of the last 2^{x-1} rows of $\mathbf{M}^x$. The first $2^{x-1}-1$ rows of $\mathbf{M}^x$ correspond to the recursively defined matrix $\mathbf{M}^{x-1}$. Finally, another copy of $\mathbf{M}^{x-1}$ is appended to the last $2^{x-1}-1$ rows of the square matrix.

The elements of every matrix $\mathbf{M}^x$ can also be reorganized into a 2^{x-1} by $2^x - 1$ rectangular matrix where all of the rows are identical, such that there are 2^{x-1} copies of x, followed by 2^{x-2} copies of $x-1$, ..., ending with 2^0 copies of 1. This directly

follows from the fact that each of the 2 $\mathbf{M}^{x-1}$ recursive submatrices is 1/2 the size of the square matrix associated with $\mathbf{M}^x$. Thus the quantity

$$\frac{\sum_{i=1}^{x} i(2^{i-1})}{2^x - 1} = \frac{2^x(x-1)+1}{2^x-1} = \frac{2^x x}{2^x-1} - 1$$

is both the average over the last row of $\mathbf{M}^x$, as well as the average over all of $\mathbf{M}^x$.

The following represents the lower triangle matrix $\mathbf{M}^3$, as well as the corresponding 2^{x-1} by $2^x - 1$ (i.e., 4 by 7) rectangular matrix.

```
1
2 2
2 2 1
3 3 3 3             3 3 3 3 2 2 1
3 3 3 3 1           3 3 3 3 2 2 1
3 3 3 3 2 2         3 3 3 3 2 2 1
3 3 3 3 2 2 1       3 3 3 3 2 2 1
```

We also note that the lower triangle matrix $\mathbf{M}^x$ has elements that are the same as the core neighbor counts for a quasi-basin of $2^x - 1$ points under both Gray and Binary that occur to the right of all possible placements of a key-transition, *except* the counts are incremented by 1.

Lemma 1: *Let $F(x)$ compute the average value over the elements of matrix $\mathbf{M}^x$:*

$$x - 1 < F(x) = \frac{2^x x}{2^x - 1} - 1 \leq x$$

Proof: From the recursive definition of $\mathbf{M}^x$ and simple induction

$$F(x) = \frac{(2^{x-1})^2 * x + 2(F(x-1)(2^{x-2})(2^{x-1}-1)}{(2^{x-1})(2^x-1)} = \frac{2^x x}{2^x-1} - 1.$$

To show that the bounds hold note:

$$F(x) = \frac{2^x x}{2^x-1} - 1 = \frac{2^x(x-1)+1}{2^x-1} > \frac{2^x(x-1)}{2^x} = x - 1$$

When $x > 1$ this implies $x < 2^x - 1$ and $F(x) = \frac{x2^x-(2^x-1)}{2^x-1} < \frac{x2^x-x}{2^x-1} = x$.
When $x = 1$ this implies $x = 2^x - 1$ and $F(x) = x = 1$.

4.2 The Matrix $\mathcal{M}^x$

We next define a new lower triangle matrix $\mathcal{M}^x$ using a constructive definition where $\mathcal{M}^1 = [0]$. Informally, the lower triangle matrix $\mathcal{M}^x$ is the same as $\mathbf{M}^x$ except the square portion of $\mathcal{M}^x$ is assigned the value $x - 1$ instead of x. Formally, the lower triangle matrix $\mathcal{M}^x$ can be decomposed into a 2^{x-1} by 2^{x-1} square matrix whose

elements are all the integer $x-1$, plus 2 identical lower triangle matrices corresponding to $\mathbf{M}^{x-1}$. The square matrix occupies the first 2^{x-1} columns of the last 2^{x-1} rows of $\mathcal{M}^x$. The first $2^{x-1}-1$ rows of $\mathcal{M}^x$ correspond to the lower triangle matrix $\mathbf{M}^{x-1}$; Finally, another copy of $\mathbf{M}^{x-1}$ is appended to the last $2^{x-1}-1$ rows of the square matrix of $\mathcal{M}$. The following is an example of $\mathcal{M}^4$.

```
      1
      2 2
      2 2 1
      3 3 3 3
      3 3 3 3 1
      3 3 3 3 2 2
      3 3 3 3 2 2 1
      3 3 3 3 3 3 3 3
      3 3 3 3 3 3 3 3 1
      3 3 3 3 3 3 3 3 2 2
->    3 3 3 3 3 3 3 3 2 2 1
      3 3 3 3 3 3 3 3 3 3 3 3
      3 3 3 3 3 3 3 3 3 3 3 3 1
      3 3 3 3 3 3 3 3 3 3 3 3 2 2
      3 3 3 3 3 3 3 3 3 3 3 3 2 2 1
```

The arrow points to row $2^{x-1}+2^{x-2}-1$ which has a special importance. The first 2^{x-2} elements have value $x-1$, and the remaining elements are identical to the last row of matrix $\mathbf{M}^{x-1}$. Recall that the last row of matrix $\mathbf{M}^{x-1}$ has the same average value as the entire $\mathbf{M}^{x-1}$ lower triangle matrix. Thus, we know the average value of the elements of this row.

The average value over all of $\mathcal{M}^x$ is greater than $F(x-1)$ but less than $x-1$, since all of the square portion of $\mathcal{M}^x$ has value $x-1$ while the submatrices of $\mathcal{M}^x$ correspond to $\mathbf{M}^{x-1}$ and therefore have average value $x-2 < F(x-1) < x-1$. We next show that the average value of the elements of every row of $\mathcal{M}^x$ numbered from 2^{x-1} to 2^x-1 is greater or equal to $F(x-1)$.

Lemma 2: Given a lower triangle matrix $\mathcal{M}^x$, $x \geq 2$ each row indexed from $2x-1$ to 2^x-1 has an average value greater than $x-2$.

Proof: The proof is constructive. By construction row $2^{x-1}+2^{x-2}$ is composed entirely of element $x-1$. The rows from $2^{x-1}+1$ to $2^{x-1}+2^{x-2}-2$ of matrix $\mathcal{M}^x$ can be constructed from row $2^{x-1}+2^{x-2}-1$ by repeatedly deleting "blocks" entirely composed of elements with value y, where y does not occur in the row under construction, and y is less than or equal to $x-2$.

The average value of row $2^{x-1}+2^{x-2}-1$ can be bounded as follows

$$x-1 > \frac{(x-1)2^{x-2}+F(x-1)(2^{x-1}-1)}{2^{x-1}+2^{x-2}-1} > x-2$$

because we can also regroup the elements and characterize row $2^{x-1}+2^{x-2}-1$ as having the value $x-1$ in the first 2^{x-2} positions with the last $2^{x-1}-1$ elements corresponding to the last row of matrix $\mathbf{M}^{x-1}$.

In general, rows $2^{x-1}+1$ to $2^{x-1}+2^{x-2}-1$ have $x-1$ as the first 2^{x-1} elements. The remaining elements corresponding to some row of the lower triangle matrix $\mathbf{M}^{x-2}$. This follows from the definition of $\mathcal{M}$ and the recursive construction $\mathbf{M}$. Row $2^{x-1}+2^{x-2}-1$ contains the last row of matrix $\mathbf{M}^{x-2}$, and therefore contains all of the blocks needed to construct all the other rows, since the last row is a slice of all the square matrices used in the recursive construction of $\mathbf{M}$. The largest value of $\mathbf{M}^{x-2}$ is $x-2$.

When we delete an element from row $2^{x-1}+2^{x-2}-1$ to create any row from $2^{x-1}+1$ to $2^{x-1}+2^{x-2}-2$ the average value of the elements in the new row must be greater than

$$\frac{(x-1)2^{x-2}+F(x-1)(2^{x-1}-1)}{2^{x-1}+2^{x-2}-1}=\frac{3(2^{x-2})(x-1)-2^{x-1}+1}{3(2^{x-2})-1}>(x-1)-\frac{2}{3}$$

This follows from the fact that deleting a below average element from a set of numbers increases the average value of the set.

Finally, the last 2^{x-2} rows $\mathbf{M}^x$ are identical to rows 2^{x-1} to $2^{x-1}+2^{x-2}-1$ except the elements are shifted right and 2^x-2 additional copies of of $x-1$ are added. Therefore, the last 2^{x-2} rows of M^x also have an average value greater than $x-2$. □

4.3 Gray Codes and Quasi-basins

Consider a contiguous set of K points that form a quasi-basin in a 1-D function. Assume these K points are intersected by a barrier. We consider all possible placements of the barrier relative to the set of points. We then compute a tight bound on the average number of neighbors that exist when the barrier is a reflection point in a Gray encoding.

The following illustration represents a quasi-basin over 7 points. Each row represents the number of neighbors for each point, such that those neighbors fall inside the quasi-basin under a Gray code, where the main reflection point of the Gray code is at the location marked by the symbol |.

```
2 3 2 3 3 3 2 |
  2 2 3 3 2 3 | 1
    1 3 2 3 3 | 2 2
      2 3 3 3 | 2 3 2
        2 3 2 | 3 3 3 2
          2 2 | 3 3 2 3 1
            1 | 3 2 3 3 2 2
              | 2 3 3 3 2 3 2
```

Theorem 2: Given a quasi-basin of size S in a 1-D function, the expected number of neighbors that fall in the quasi-basin under Gray code is greater than $\lfloor \log_2(S) \rfloor - 1$.

Proof: For $S=2^{x-1}$ to 2^x-1 we note that $\lfloor \log_2(S) \rfloor - 1 = x-2$.

We consider all possible placements of the Gray code reflection key-transition, and average over all possible neighborhood arrangements. This means that all probabilities

in the expected value calculation are uniform. The neighborhood structure is symmetric around the key-transition; we can therefore consider only the lower triangle matrix which lies to the right of the key-transition.

When $S = 2^x - 1$ we can use the matrix $\mathcal{M}^x$ to bound the expected number of neighbors for every point in the quasi-basin on one side of the main barrier/reflection for all possible placements of the reflection points.

Let S vary from 2^{x-1} to $2^x - 1$. A group of 2^k points adjacent to a reflection must have at least k core neighbors under Gray code; the square submatrix of $\mathcal{M}^x$ counts only these core neighbors. The lower triangle submatrices corresponding to $\mathbf{M}^{x-1}$ counts core neighbors, plus 1 more in each position. Under Gray code, there is at least one additional neighbor if there is another point in the space in a position symmetric around a reflection. This is the case for all elements in the positions that correspond to the two $\mathbf{M}^{x-1}$ lower triangle matrices. There are at least 2^{x-1} points in the quasi-basin. The square submatrix of $\mathbf{M}^{x-1}$ has 2^{x-2} core neighbors, so there must be an additional 2^{x-2} neighbors that fall either to the left or right of a reflection to either side of the 2^{x-2} core neighbors; thus each point in the square submatrix has one additional neighbor. The lower triangle submatrices of $\mathbf{M}^{x-1}$ also have one additional neighbor across the reflection that occurs immediately to the left of these lower triangle submatrices.

When $2^{x-1} \leq S \leq 2^x - 1$ we compute a bound on the average over the first S rows of matrix $\mathcal{M}^x$. The submatrix of $\mathbf{M}^{x-1}$ makes up the first $2^{x-2} - 1$ rows of matrix $\mathcal{M}^x$; by lemma 1, the average value of the elements in $\mathbf{M}^{x-1}$ is greater than $x - 2$. We select the next $S - (2^{x-2} - 1)$ rows that are needed from matrix $\mathcal{M}^x$. From lemma 2, the average value over each of these rows is greater than $x - 2$.

Thus, for any point in a quasi-basin of size S, the expected number of neighbors that also fall in the quasi-basin is greater than $\lfloor \log_2(S) \rfloor - 1$. □

Corollary: *Given a quasi-basin that spans $1/Q$ of the search space and a reference point R that falls in the quasi-basin, the majority of the neighbors of R under a reflected Gray code representation of a search space of size $N = 2^L$ will be subthreshold in expectation when $\lfloor \log(N/Q) \rfloor - 1 > \log(Q) + 1$.*

This result has strong implications for local search. It means that under the conditions just outlined, we can expect a majority of the neighbors that are sampled under local search using a Hamming distance-1 Gray encoded bit representation neighborhood to also be below threshold when searching from a subthreshold point. It also follows from these observations that the percentage of below threshold neighbors increases at higher precision.

5 Binary Codes and Quasi-basins

We have already noted that Binary encodings have the same core neighborhood as Gray. But what about additional, non-core neighbors?

We start with the special case where there are exactly $2^x - 1$ elements in the quasi-basin. The number of neighbors when $x = 3$ and $2^x - 1 = 7$ is illustrated by the following example.

```
1 2 3 2 3 3 3 |
  2 2 2 2 3 3 | 0
    1 2 2 2 3 | 1 1
      2 2 2 2 | 2 1 1
        1 1 2 | 2 2 2 2
          1 1 | 3 2 2 2 1
            0 | 3 3 2 2 2 2
              | 3 3 3 2 3 2 1
```

We will again work with the lower triangle form of the matrix. This lower triangle matrix can still be recursively decomposed but the elements in the topmost recursive matrix and the rightmost recursive matrix differ by 1 in all positions. Furthermore, the square matrix has the value $x-1$ in all positions except in its own lower triangle: the value x appears in the lower triangle of the square matrix. The reflected neighbors are gone, which is why the topmost recursive matrix is 1 less in every position. Each element corresponding to a position in the rightmost recursive lower triangle matrix has a neighbor at distance 2^{x-1} in the lower minor triangle of the square matrix.

Lemma 4:

For a quasi-basin of exactly $2^x - 1$ elements, the number of neighbors that are in the quasi-basin is exactly $x-1$ under a Binary encoding.

Proof:

The proof is by induction. This is true by inspection for $x = 1, 2$ and 3.

We will again use a recursive lower triangle representation that decomposed into a square matrix and two recursively defined lower triangle submatrices. These count the core neighbors, plus those additional neighbors that occur due to the Binary encoding.

Assume the lemma is true for case $x-1$. Then the topmost recursive matrix represents the lower triangle matrix associated with $x-1$ and has an average value per element of exactly $x-2$ by the inductive hypothesis; the topmost recursive matrix has $2^{x-2}(2^{x-1}-1)$ elements. The rightmost recursively defined lower triangle submatrix has an average value per element of exactly $x-1$. This includes the core neighbors, plus one additional neighbor, since any point more than 2^{x-1} positions to the right of the key transition must have a (non-core) neighbor in the square submatrix.

The square submatrix has $2^{x-1}2^{x-1}$ elements and $2^{x-2}(2^{x-1}-1)$ in its own lower minor triangle, which connect to an element in the rightmost recursively defined lower triangle submatrix. Adding these together yields.

$$2^{x-2}(2^{x-1}-1)(x-2)+2^{x-2}(2^{x-1}-1)(x-1)+2^{x-1}2^{x-1}(x-1)+2^{x-2}(2^{x-1}-1)$$
$$= [(2^{x-1})^2 + 2^{x-1}(2^{x-1}-1)](x-1)$$

Therefore, since the number of elements in the recursively defined matrix is $(2^{x-1})^2 + 2^{x-1}(2^{x-1}-1)$ the average value per element is exactly $x-1$. □

Theorem 3: Given a quasi-basin of size S in a 1-D function, the expected number of neighbors that fall in the quasi-basin under Binary code is greater than $\lfloor \log_2(S) \rfloor - 1$.

Proof: For $S = 2^{x-1}$ to $2^x - 1$ we note that $\lfloor log_2(S) \rfloor - 1 = x - 2$.

We again consider all possible positions in the quasi-basin for which the number of neighbors is being computed, and all possible placements of the key-transition barrier. Thus, all probabilities in the expected value calculation are uniform.

If $S = 2^{x-1} - 1$ (this is one element smaller than is allowed for S), the number of neighbors is precisely $x - 2$ on average (lemma 4).

We will let S vary from 2^{x-1} to $2^x - 1$. A group of 2^i points adjacent to a key-transition must have at least i core neighbors; the square submatrix of $\mathcal{M}^x$ counts only these core neighbors. For rows 2^{x-1} to $2^x - 1$ the lower triangle submatrix corresponding to M^{x-1} which is right of the square submatrix is such that each element must have one neighbor at a distance of 2^{x-1} to the right, since the square submatrix of $\mathcal{M}^x$ is of size 2^{x-1}.

When $2^{x-1} \leq S \leq 2^x - 1$ we select the $S - 2^{x-2} - 1$ rows that are needed from matrix $\mathcal{M}^x$; from lemmas 1 and 2, all of these rows have average value greater than $x-2$. □

Corollary: *Given a quasi-basin that spans $1/Q$ of the search space and a reference point R that falls in the quasi-basin, the majority of the neighbors of R under a Binary representation of a search space of size $N = 2^L$ will be subthreshold in expectation when $\lfloor \log(N/Q) \rfloor - 1 > \log(Q) + 1$.*

6 Discussion

A steepest ascent local search algorithm currently at a subthreshold point can only move to an equal or better point which must also be subthreshold. And as precision increases, the number of subthreshold neighbors also increases, since $\lfloor \log(N/Q) \rfloor - 1$ increases while Q remains constant. This assumes the quasi-basin is not divided by increasing the precision.

The above analysis would need to hold for each dimension of a multidimensional search space, but these results suggest there are very general conditions where a steepest ascent bit-climbing algorithm using either a Binary or a Gray code representation can display subthreshold-seeking behavior. This also assumes the search algorithm can absorb the start-up costs of locating a subthreshold starting point.

Whitley, Rowe and Bush [WRB04] proposed a subthreshold bit climber called LS-SubT that uses a Gray Code representation. LS-SubT first samples 1000 random points, and then climbs from the 100 best of these points. In this way, LS-SubT estimates a threshold value and attempts to spend a majority of its time in the best 10 percent of the search space.

Experimental results on a number of benchmarks indicate that LS-SubT sometimes produces statistically significant better solution quality compared to steepest ascent bit climbing local search. LS-SubT never produces statistically significant worse performance than steepest ascent bit climbing local search. The data suggests two additional observations about subthreshold-seeking behavior [C01] [WRB04]. First, the sampling used by LS-SubT results in a higher proportion of subthreshold points compared to steepest ascent bit climbing local search. Second, a larger proportion of subthreshold

neighbors are sampled for searches using higher precision. At 10 bits of precision per parameter, LS-SubT sampled subthreshold points more than 55 percent of the time. At 10 bits of precision, LS-SubT also used statistically significanbt fewer evaluations than steepest ascent local search. At 20 bits of precision per parameter, and at least 80 percent of the points samples by LS-SubT were subthreshold.

The locality results presented in this paper can perhaps be generalized to cover both next ascent bit climbers such as Random Bit Climbing (RBC) as well as Quad Search. While Quad Search looks different, it progressively looks closer and closer to the current best solution. Consider that RBC looks at all L bit flips before testing any bit again. Under RBC, checking the highest order bit samples a point in the other "half" of Hamming space (note this point is not necessarily far away under Gray code). Assuming that RBC does not find an improving move, the bits could be re-sorted so that high-order bits are sampled first, in effect reducing the search space by half after each bit flip. Under Quad Search, two points are checked in the other half of the space before the search space is reduced by half. Again assuming that Quad Search does not find an improving move, it checks at most $2L$ points while reducing the search space by half after every 2 bit flips. From this perspective, RBC and Quad Search have similar locality: the percentage of points that are close the the current best solutions should be nearly identical in expectation (except the modified Quad Search will potentially skip some evaluations.) This suggests that the only major adjustment needed to generalize the locality results for steepest ascent is to account for the fact that the current solution is changing location in both next ascent and Quad Search. We conjecture this makes no difference in the expected number of neighbors that fall in the quasi-basin.

7 Conclusions

Experiments have shown that a hybridization of Quad Search with a steady state genetic algorithm finds much better solutions on difficult benchmark problems than hybrid genetic algorithms that are combined with steepest ascent local search or RBC (Random Bit Climbing). The apparent reason for this is that Quad Search very quickly converges to good solutions.

The modified Quad Search iteratively optimizes in each dimension so that it is guaranteed to converge to a locally optimal solution *in that dimension* before it moves to the next dimension. As with line search, one pass of the Quad Search is not guaranteed to converge to a locally optimal solution. On the other hand, this may make Quad Search more stochastic in its exploration of the search space compared to steepest ascent and next ascent.

Quad Search exploits unique properties of Gray codes and there would appear to be no direct analogy to Quad Search using Binary representations that provides the same generality and flexibility.

This paper also looks at locality, showing that most neighbors samples under both Gray and Binary representations are very close to the current reference point around which the neighborhood is defined. If that reference point R is in a quasi-basin that spans $1/Q$ of the search space then the majority of the neighbors of R under a reflected Gray code representation or a Binary representation of a search space of size $N = 2^L$

will be subthreshold in expectation when $\lfloor \log(N/Q) \rfloor - 1 > \log(Q) + 1$. This outlines conditions that will allow a steepest ascent bit climber to spend a majority of its time below threshold, and thus out perform random search. We conjecture the same results also hold for Quad Search.

References

[B̈96] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, 1996.

[BER76] James R. Bitner, Gideon Ehrlich, and Edward M. Reingold. Efficient Generation of the Binary Reflected Gray Code and Its Applications. *Communications of the ACM*, 19(9):517–521, 1976.

[C01] S. Christensen and F. Oppacher (2001). What can we learn from No Free Lunch? In *GECCO-01*, pages 1219–1226. Morgan Kaufmann, 2001.

[CS88] R. Caruana and J. Schaffer. Representation and Hidden Bias: Gray vs. Binary Coding for Genetic Algorithms. In *Proc. of the 5th Int'l. Conf. on Machine Learning*. Morgan Kaufmann, 1988.

[Dav91] Lawrence Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[RW97] S. Rana and D. Whitley. Representations, Search and Local Optima. In *Proceedings of the 14th National Conference on Artificial Intelligence AAAI-97*, pages 497–502. MIT Press, 1997.

[RWB04] J. Rowe, D. Whitley, L. Barbulescu, and J.P. Watson. Properties of Gray and Binary Representations. *Evolutionary Computation*, 12:in press, 2004.

[WGW03] D. Whitley, D. Garrett, and J. Watson. Genetic Quad Search. In *GECCO 2003*, pages 1469–1480. Springer, 2003.

[Whi99] D. Whitley. A Free Lunch Proof for Gray versus Binary Encodings. In *GECCO-99*, pages 726–733. Morgan Kaufmann, 1999.

[WM95] 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.

[WRB04] D. Whitley, J. Rowe, and K. Bush. Subthreshold Seeking Behavior and Robust Local Search. In *GECCO 2004*, pages v2:282–293. Springer, 2004.