# New Insights about No Free Lunch

Darrell Whitley
Colorado State AI Lab
Colorado State University



**NFL: No Free Lunch**

*All search algorithms are equivalent when compared over all possible discrete functions.*

Wolpert, Macready (1995)
No free lunch theorems for search. Santa Fe Institute.

Radcliffe, Surry (1995)
Fundamental Limitations on Search Algorithms: Springer Verlag LNCS 1000.

**No Free Lunch for Gray and Binary**

*All search algorithms are equivalent when compared over all possible representations.*

## Variations on No Free Lunch

For ANY measure of algorithm performance:

The aggregate behavior of any two search algorithms is equivalent when compared all possible discrete functions.

The aggregate behavior of ALL possible search algorithms is equivalent when compared over any two discrete functions.

At each distinct "iteration" of search
the aggregate behavior of all possible search algorithms is IDENTICAL at each and every iteration.



## Variations on No Free Lunch

Consider any algorithm $A_i$ applied to function $f_j$.

On($A_i, f_j$) outputs the order in which $A_i$ visits the elements in the codomain of $f_j$. For every pair of algorithms $A_k$ and $A_i$ and for any function $f_j$, there exist a function $f_l$ such that

$$On(A_i, f_j) \equiv On(A_k, f_l)$$

Consider a "BestFirst" local search with restarts.

Consider a "WorstFirst" local search with restarts.

For every $j$ there exists an $l$ such that

$$On(BestFirst, f_j) \equiv On(WorstFirst, f_l)$$

ENUMERATION is a search algorithm.

Thus, No Free Lunch implies that on average,
no search algorithm is better than enumeration.

Furthermore, because bias in search algorithms causes them to focus the search, most are prone to resampling.

If resampling is considered,
"focused" search algorithms are WORSE than enumeration

NFL IGNORES RESAMPLING



An algorithm is modeled as a permutation
representing the order in which new points are tested.

Behavior is defined in terms of the evaluation function output
which defines the co-domain of the function.

Assume that one is given a fixed set of co-domain values.
Set of Functions = Set of Permutations.

| BEHAVIORS | | FUNCTIONS | |
|---|---|---|---|
| A1: | 1 2 3 | F1: | A B C |
| A2: | 1 3 2 | F2: | A C B |
| A3: | 2 1 3 | F3: | B A C |
| A4: | 2 3 1 | F4: | B C A |
| A5: | 3 1 2 | F5: | C A B |
| A6: | 3 2 1 | F6: | C B A |



Assume $(A > B)\&(B > C)$.
Take 2 steps, return the maximum found.

| | F1 | F2 | F3 | F4 | F5 | F6 |
|---|---|---|---|---|---|---|
| A1 | A | A | A | B | A | B |
| A2 | A | A | B | A | B | A |
| A3 | A | A | A | B | A | B |
| A4 | B | B | A | A | A | A |
| A5 | A | A | B | A | B | A |
| A6 | B | B | A | A | A | A |

**Theorem:**

NFL holds for a set of functions IFF
the set of functions form a permutation set.

The “Permutation Set” is the closure of a set
of functions with respect to a permutation operator.
(Schmacher, Vose and Whitley–GECCO 2001).

```
F1:  0 0 1 2       F7:  0 2 0 1
F2:  0 1 0 2       F8:  0 2 1 0
F3:  1 0 0 2       F9:  1 2 0 0

F4:  0 0 2 1      F10:  2 0 0 1
F5:  0 1 2 0      F11:  2 0 1 0
F6:  1 0 2 0      F12:  2 1 0 0
```



OBSERVATION: The Union of Permutation Sets is also a Permutation Set.
The sampling probability can be different across Permutation Sets.

Sampling Need not be Uniform

```
F1:  A B C   12/100        F1:  0 0 0 1   7/100
F2:  A C B   12/100        F2:  0 0 1 0   7/100
F3:  B A C   12/100        F3:  0 1 0 0   7/100
F4:  B C A   12/100        F4:  1 0 0 0   7/100
F5:  C A B   12/100
F6:  C B A   12/100
```

## Machine Learning and NFL

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | | | 1 |
| 0 | | | 0 |
| 0 | 0 | 0 | 0 |

```
L1   ALL   HD          L2   ALL   HD
==   ========          ==   ========
     00    0                00    1
00   01    1           10   01    2
     10    1                10    0
     11    2                11    1
```



**Theorem:**
Given a finite set of N unique co-domain values, NFL hold over a set of N! functions where the average description length is O(N log N).

**Sketch of Proof:**
Construction a Binary Tree with N! leaves. Each leaf represents one of the N! functions. To just label each function requires log(N!) bits. Each label has average length log(N!) = O(N log N).
Note enumeration also has cost O(N log N).

**Corollary:**
If a fixed fraction of the co-domain values are unique, the set of N! functions where NFL holds has average description length O(N log N).

NFL holds over sets with 1 member.

```
F = 0 0 0 0
```

NFL holds over needle-in-a-haystack functions.

```
F1 = 0 0 0 1
F2 = 0 0 1 0
F3 = 0 1 0 0
F4 = 1 0 0 0
```



The set of Binary strings is a permutation set

```
0 0 0 0                      1 1 1 1

0 0 0 1       0 0 1 1        1 1 1 0
0 0 1 0       0 1 0 1        1 1 0 1
0 1 0 0       1 0 0 1        1 0 1 1
1 0 0 0       0 1 1 0        0 1 1 1
              1 0 1 0
              1 1 0 0
```

Let $P(F)$ compute the permutation closure of $F$, where $F$ is a set of functions.

Let $K = |P(F)|$.

Then the average description length needed to distinguish the members of that set is $lg(K)$.

If $lg(K)$ is exponential, then the permutation set is $uncompressible$.

If $lg(K)$ is polynomial, then the permutation set is $compressible$.



QUESTION:
How should we evaluate search algorithms?

Let $\beta$ represent a set of benchmarks. $P(\beta)$ is the permutation closure over $\beta$.

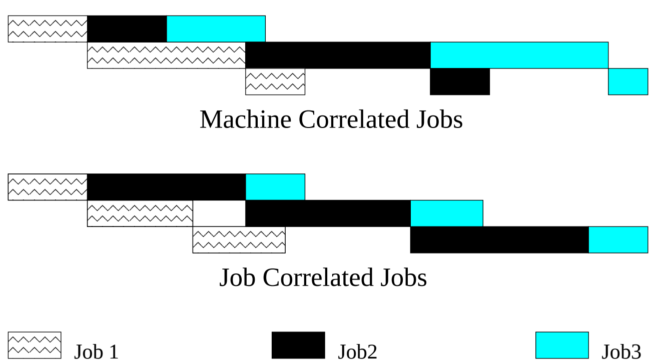
*If algorithm* **S** *is better than algorithm* **T** *on* $\beta$
*THEN* **T** *is better than* **S** *on* $P(\beta) - \beta$.

NO FREE LUNCH is not proven to hold over the class of problems in NP unless we prove that $P \neq NP$. If $P = NP$ then there are more efficient algorithms than RANDOM SEARCH.

NO FREE LUNCH does not hold over the class of problems in NP that have ratio bounds which can be exploited by branch and bound algorithms.

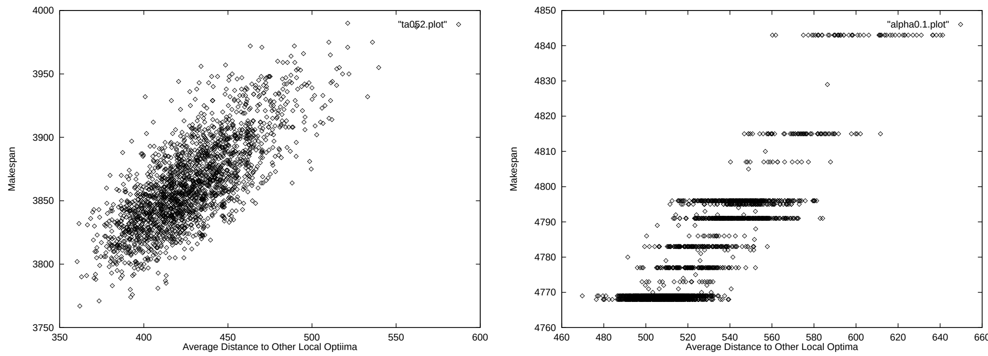
Does NFL hold for “rich” problems/languages problems that have polynomial descriptions that we want to solve in practice?



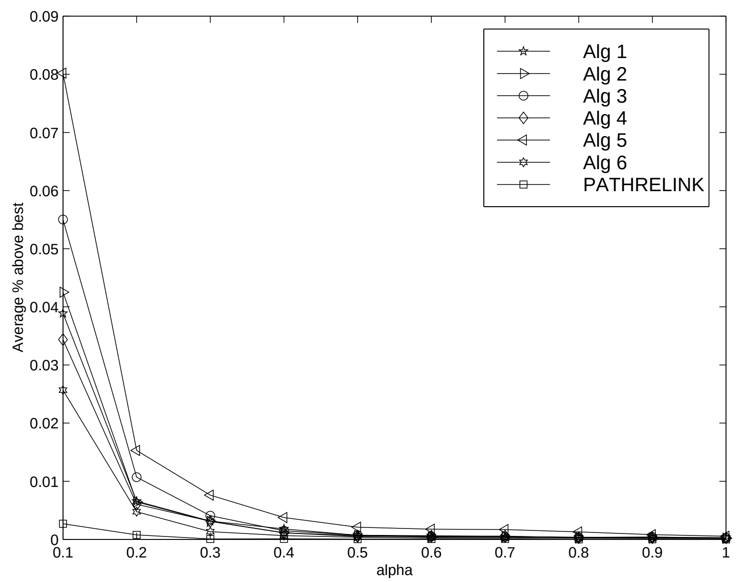



The PERMUTATION FLOWSHOP SCHEDULING PROBLEM.

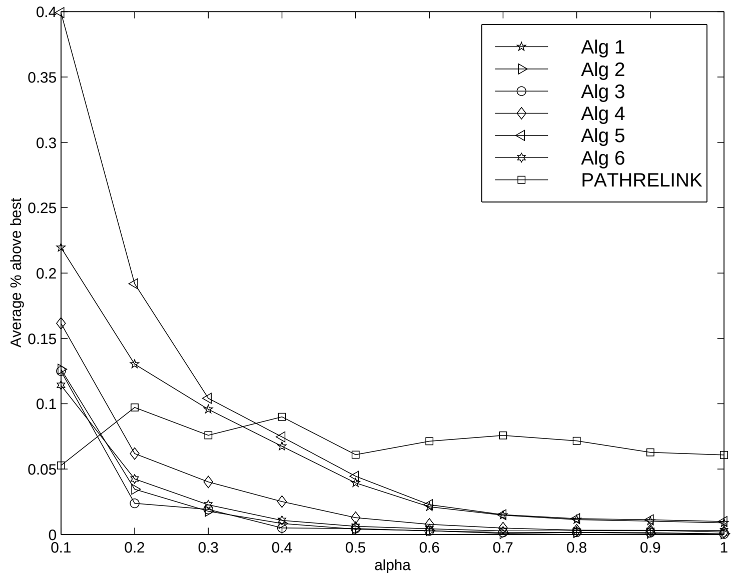
Benchmark are typically generated randomly. Real-world problems may have correlated structure. Job could be *machine correlated* or *job correlated*.






JOB CORRELATED PROBLEMS. Performance of optimization algorithms. The degree of randomness is indicated along the x-axis, while the deviation from the best-known solution is indicated along the y-axis.




MACHINE CORRELATED PROBLEMS. Performance of optimization algorithms. The degree of randomness is indicated along the x-axis, while the deviation from the best-known solution is indicated along the y-axis.



S. Christensen and F. Oppacher
*What can we learn from No Free Lunch?* GECCO 2001

A SUBMEDIAN-SEEKER Type Algorithm

1. Evaluate a sample of points and estimate median(f).
2. If $f(x_i) < median(f)$ then sample a neighbor of $x_i$.
   Else sample a new random point.
3. Repeat step 2 until half of space is explored.

Assume $f$ is 1-dimensional, a bijection, and we know $median(f)$.

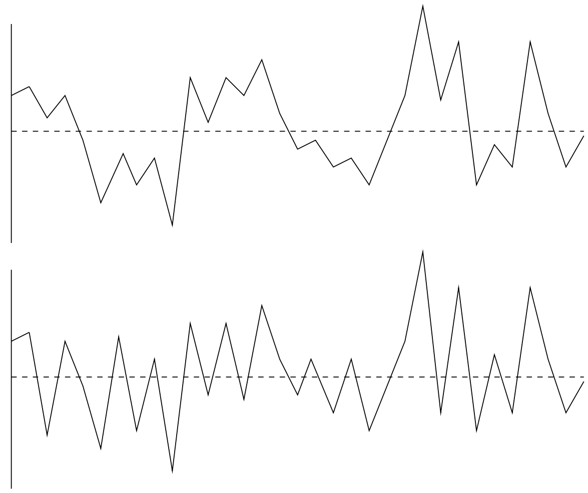
Let $M(f)$ measures the number of submedian values of $f$ that have *supermedian successors*.

There exists $M_{crit}$ such that when $M(f) < M_{crit}$ SubMedian-Seeker is better than random search.

SUBMEDIAN-SEEKER beats random enumeration when:

1. $f$ is a uniformly sample polynomial of degree at most $k$ and $M_{crit} > k/2$
2. $f$ is a truncated Fourier series of at most $k$ harmonics uniformly sampled over [0,1) at $n$ locations and $M_{crit} > k/2$
3. Each extremum of $f$ is represented by at least 6 points on average





## Structure is Important

Random Number Generators produce functions that are in some restricted sense compressible. But they are designed to have minimal structure.

Consider "WorstFirst" local search again.
For every $j$ there exists an $l$ such that

$$On(BestFirst, f_j) \equiv On(WorstFirst, f_l)$$

There are "structured functions" that do not fit our usual notion of being "searchable."



NO FREE LUNCH and REPRESENTATION

Radcliffe, Surry (1995) Fundamental Limitations on Search Algorithms: Springer Verlag LNCS 1000.

The behavior of any two algorithms are identical over all possible representations of a single function.

"NO-FREE-LUNCH-like" results

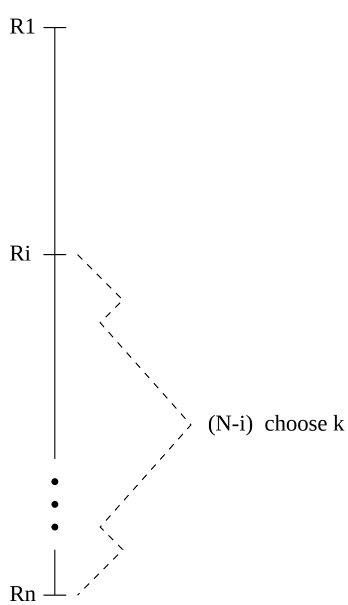
The behavior of any two algorithms are identical over over the set of Gray and the set of Binary representations over all possible functions.

# Counting Local Optima

The probability that string $i$ is a local minimun under an arbitrary transformation of a k-neighborhood search space is:

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



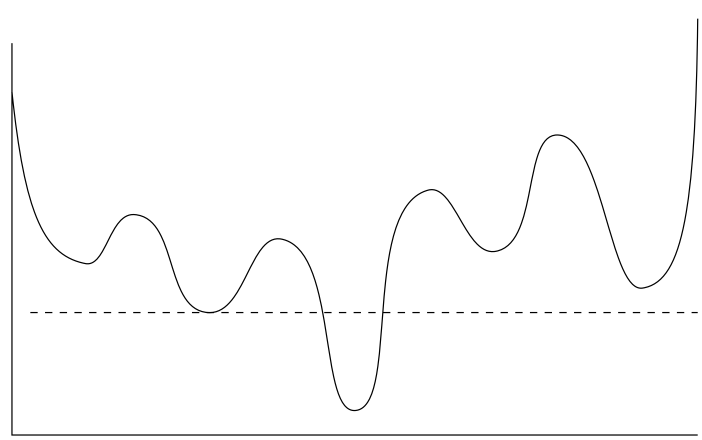


The average number of local optima over all possible representations using a k-neighbor search:

$$\mu(N,k) = \sum_{i=1}^{N-k} P(i) \tag{2}$$

$$\mu(N,k) = N/(k+1) \tag{3}$$



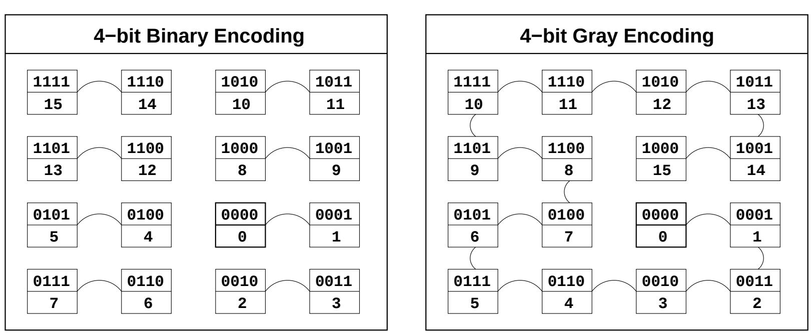


| BINARY | | GRAY |
|---|---|---|
| 000 | 0 | 000 |
| 001 | 1 | 001 |
| 010 | 2 | 011 |
| 011 | 3 | 010 |
| 100 | 4 | 110 |
| 101 | 5 | 111 |
| 110 | 6 | 101 |
| 111 | 7 | 100 |

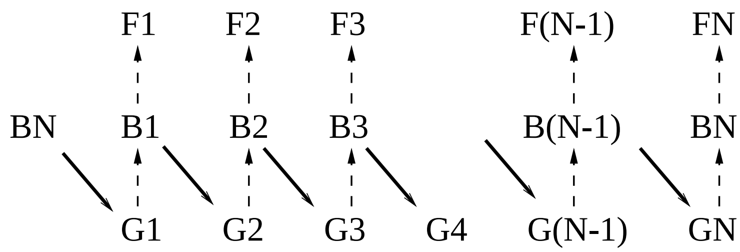
---

| | Gray Matrix | Degray Matrix |
|---|---|---|
| 3-bits | $\begin{vmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{vmatrix}$ | $\begin{vmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{vmatrix}$ |
| 5-bits | $\begin{vmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{vmatrix}$ | $\begin{vmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{vmatrix}$ |





”NO-FREE-LUNCH-like” results hold over
very small sets of functions for Gray and Binary representations.




The length of this “chain” is at most 2L.



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| R1: | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| R2: | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
| R3: | 000 | 001 | 010 | 011 | 101 | 100 | 111 | 110 |
| R4: | 000 | 001 | 011 | 010 | 111 | 110 | 100 | 101 |
| R5: | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |

Consider the integer-adjacency neighborhood.

1, 2, 3, 4, 5, 6, 7, 8, ... N-3, N-2, N-1, N

We consider a WRAPPING Neighborhood
where 1 and N are neighbors.

(We can also consider a NON-WRAPPED Neighborhood,
where 1 and N are not neighbors).

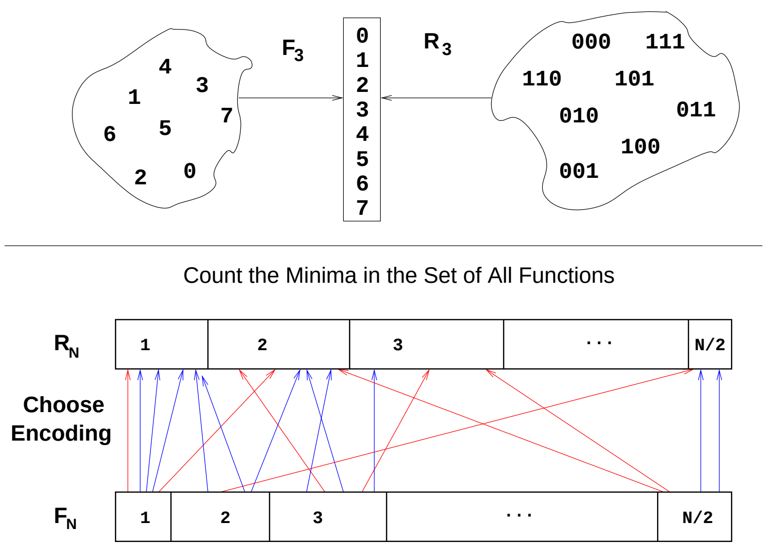


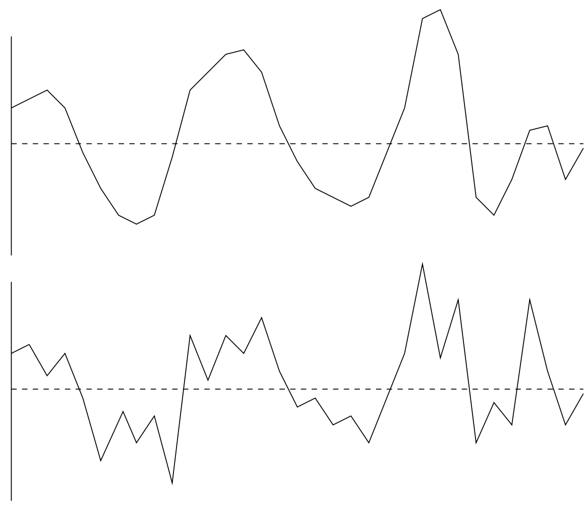
| FOR WRAPPING FUNCTIONS | | | |
|---|---|---|---|
| K | #F<br>K Min | # of Min<br>Gray | # of Min<br>Binary |
| 1 | 512 | 512 | 1,024 |
| 2 | 14,592 | 23,040 | 27,776 |
| 3 | 23,040 | 49,152 | 48,896 |
| 4 | 2,176 | 7,936 | 2,944 |
| Sum | 40,320 | 80,640 | 80,640 |

| MINI-MAX: WRAPPING | | | |
|---|---|---|---|
| K | Gray Wins | Binary Wins | Ties |
| 1 | 448 | 0 | 64 |
| 2 | 6752 | 2288 | 5552 |
| 3 | 6720 | 6592 | 9728 |
| 4 | 0 | 2160 | 16 |
| Sum | 13,920 | 11,040 | 15,360 |



Generating the Set of All Functions







A SubThreshold-Seeker

1. Evaluate a sample of points and estimate a $threshold(f)$.
2. Pick point $x < threshold(f)$.
3. If $f(x) < threshold(f)$ then set $x = x + 1$ and $y = x - 1$;
   Else sample a new random point.
4. While $f(x) < threshold(f)$ set $x = x + 1$;
5. While $f(y) < threshold(f)$ set $y = y - 1$;
6. If stopping-conditions not met, goto 2.

Define a *quasi-basin* as a contiguous set of points below threshold. Let $\alpha$ define a threshold presenting some fraction of the search space. Suppose there are $B$ quasi-basins each containing at least $M$ points.

**Theorem:** *Suppose that Subthreshold-Seeker is used to find $B$ quasi-basins each containing at least $M$ points. Forall $\alpha < 1/2$ subtheshold-seeker beats random search if $M > \sqrt{\frac{NH(B-1)}{B}}$.*

$\sqrt{\frac{NH(B-1)}{B}}$ does not reference $\alpha$ because $M$ is derived from $\alpha$.



What about a simple bit climber using Gray Code?

**Theorem:** *Given a quasi-basin that spans $1/Q$ of a search space of size $N$ 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*

$$\lfloor(log(N/Q))\rfloor - 1$$

**Corollary:** *Given a quasi-basin below theshold $\alpha$ that spans $1/Q$ of the search space and a reference point $R$ that fall in the quasi-basin, the majority of the neighbors of $R$ under a reflected Gray code representation of a search space of size $N$ will also be subthreshold in expectation when*

$$\lfloor(log(N/Q))\rfloor - 1 > \log(Q) + 1$$

This means that a simple “local search” bit climber can beat random enumeration when restarted from a subthreshold points as long as on average

$$\lfloor (log(N/Q)) \rfloor - 1 > \log(Q) + 1$$

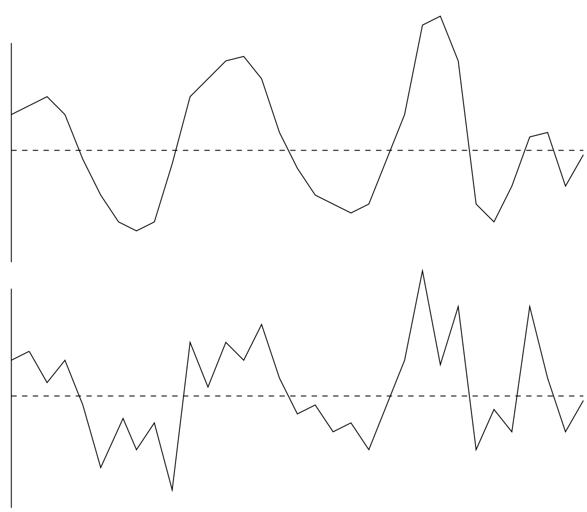
Let $N = 2^{100}$ and assume we want to largely sample a quasi-basin that spans $1/billon^{th}$ of the space.

$$\lfloor (log(2^{100}/2^{30})) \rfloor - 1 > \log(2^{30}) + 1$$

$$69 > 31$$

NOTE: An increase in precision increases $\lfloor (log(N/Q)) \rfloor - 1$ but does not increase $\log(Q) + 1$.





|  |  | *10 bit Precision* |  |  |  | *20 bit Precision* |  |  |
|---|---|---|---|---|---|---|---|---|
| Func | ALG | Mean | Sub | Evals |  | Mean | Sub | Evals |
| ackley | R-LS | 0.18 | 62.4 | 19371 |  | 0.0001 | 75.1 | 77835 |
|  | SubT | 0.18 | 79.7 | 16214† |  | 0.0001 | 89.9 | 73212† |
| grie- | R-LS | 0.010 | 59.5 | 13412 |  | 0.0045 | 80.3 | 66609 |
| wangk | SubT | 0.005 | 80.1 | 9692† |  | 0.0049 | 90.0 | 59935† |
| rana | R-LS | -49.6 | 49.5 | 22575 |  | -49.76 | 74.2 | $3\times10^6$ |
|  | SubT | -49.4 | 57.6 | 19453† |  | -49.83 | 85.0 | $3\times10^6$ |

Table 1: Local Search Results averaged over 30 runs. Threshold = 10 percent. The † denotes statistical significance.



# References

[1] S. Christensen and F. Oppacher. *What can we learn from No Free Lunch*. GECCO 2002.

[2] J. Culberson. On the Futility of Blind Search. *Evolutionary Computation*, 6(2):109–127, 1999.

[3] S. Droste and T. Jansen and I. Wegener. Perhaps not a free lunch but at least a free appetizer. *GECCO*, 1999.

[4] S. Droste and T. Jansen and I. Wegener. Optimization with randomized search heuristics; the (A)NFL theorem, realistic scenarios and diffi cult functions. *Theoretical Computer Science*, 2002 (In Press).

[5] T. English. Practical implications of new results in conversation of optimizer performance. *Parallel Problem Solving from Nature*, 2000.

[6] T. English. Information is Conserved in Optimization. *IEEE Trans Evolutionary Computation*.

[7] N.J. Radcliffe and P.D. Surry. Fundamental limitations on search algorithms: Evolutionary computing in perspective. *Lecture Notes in Computer Science 1000*. Springer-Verlag, 1995.

[8] C. Schumacher. *Fundamental Limitations of Search*. PhD thesis, University of Tennessee, Department of Computer Sciences, Knoxville, TN, 2000.

[9] C. Schumacher and M. Vose and D. Whitley. *The No Free Lunch and Problem Description Length*. *GECCO* 2001.

[10] D. Whitley. Functions as permutations: regarding no free lunch, walsh analysis and summary statistics. *Parallel Problem Solving from Nature, 6*, 2000.

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

[12] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Trans Evolutionary Computation*, 4:67–82, 1997.