Practical Guidelines for Using Evolutionary Algorithms

Darrell Whitley
Colorado State AI Lab
Colorado State University

A Sample Set of Evolutionary Algorithms

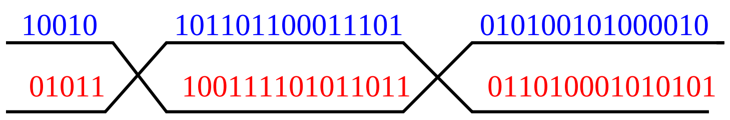
- **Simple Genetic Algorithm**: Holland/Goldberg
- **Genitor, Steady-State GAs**: Whitley
- **CHC**: Eshelman
- **Evolution Strategies**: Schwefel/Rechenburg
- **CMA Evolution Strategies**:Hansen, Ostermeier
- **Parallel Genetic Algorithms**
  - **Island Model Genetic Algorithms**
  - **Cellular Genetic Algorithms**
- **Other Algorithms:**
  - **Tabu Search**
  - **Pattern Search, Mesh Adaptive Direct Search**

The Simple Genetic Algorithm (with Elitism)

RECOMBINATION: Let the following two binary strings represent an encoding of 5 parameters in a parameter optimization problem.

1001010 1101100 0111010 1010010 1000010

0101110 0111101 0110110 1101000 1010101

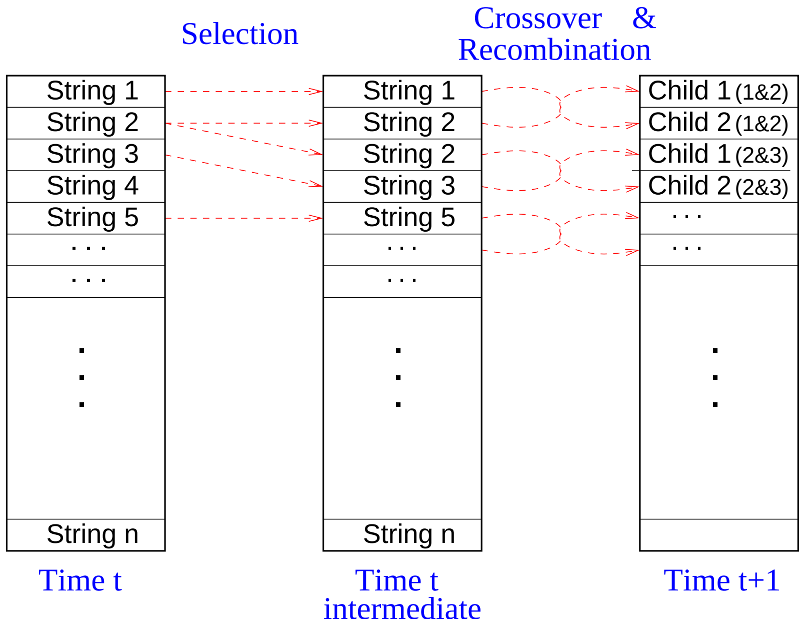


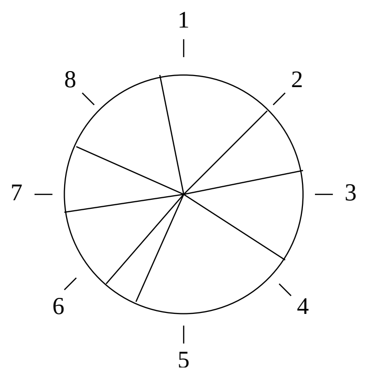
Which Produces the Offspring

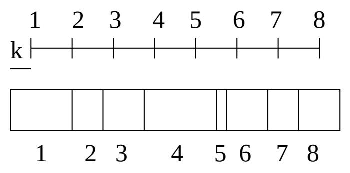
01011101101100011101011010001010101

10010100111101011011010100101000010

SIMPLE GENETIC ALGORITHM MODEL

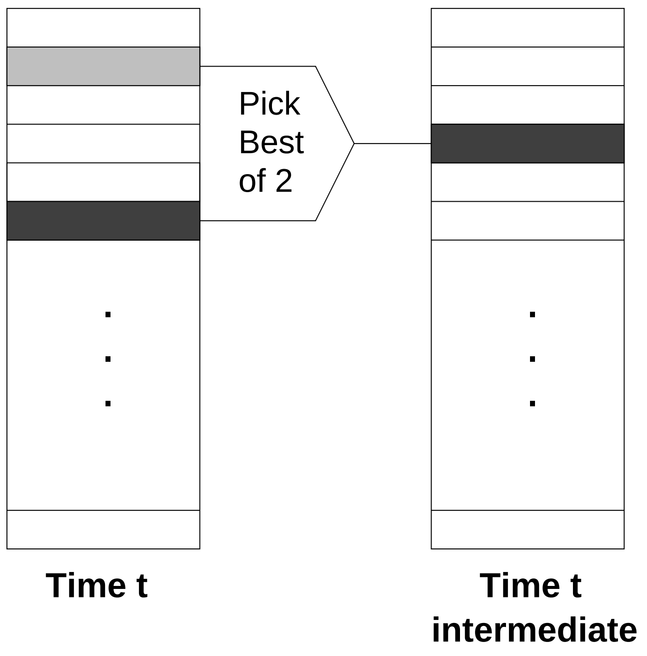






Universal Stochastic Sampling, Roulette Wheel Selection

TOURNAMENT SELECTION



Tournament Selection with Variance Reduction

Assume population size $= K$.
Let $\pi_k(i)$ be the i^{th} element of a random permutation π_k.

```
For $i = 1$ to $K$
  Compare populaton member $i^{th}$ against member $\pi_k(i)$.
  Keep the best.
```

```
                                Tournaments
   Every individual is in         1 vs 3
   EXACTLY 2 Tournaments          2 vs 7
                                  3 vs 6
                                  4 vs 5
                                  5 vs 1
                                  6 vs 2
                                  7 vs 4
```

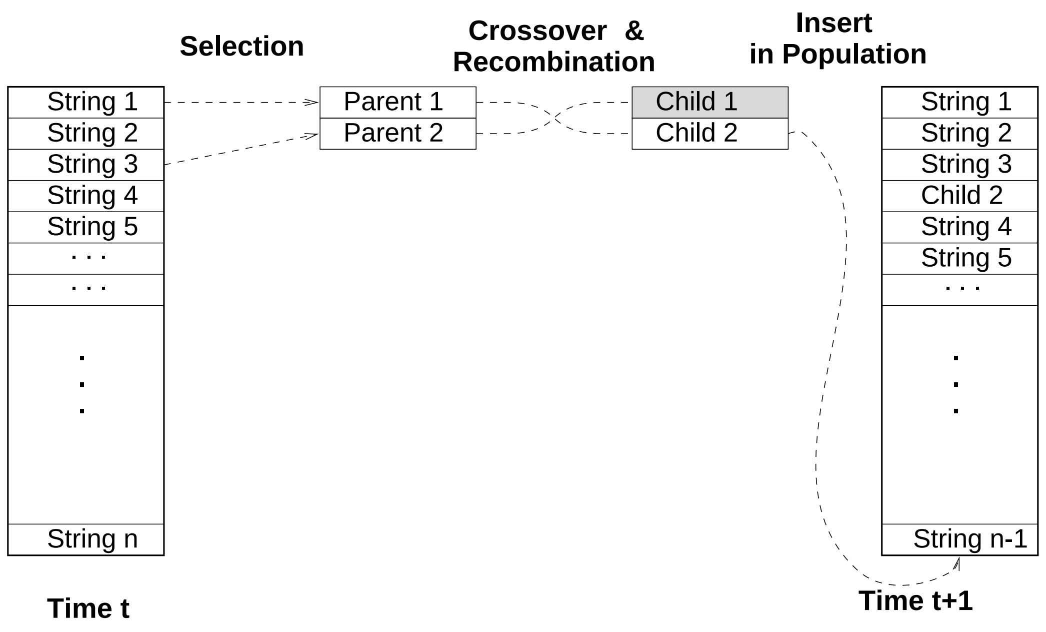
Genitor: A "Steady State" GA

- Rank Based Selection
- Two Point Crossover with Reduced Surrogates
- Randomly Choose One Offspring, Mutate
- Insert and Displace Worst

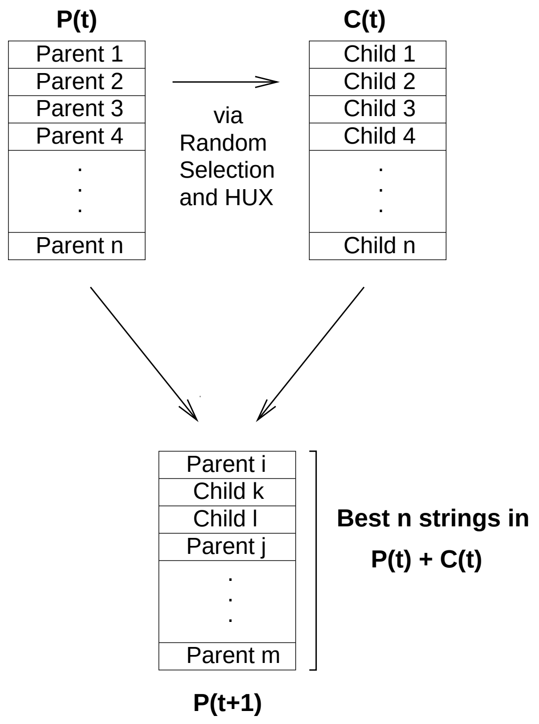
CHC

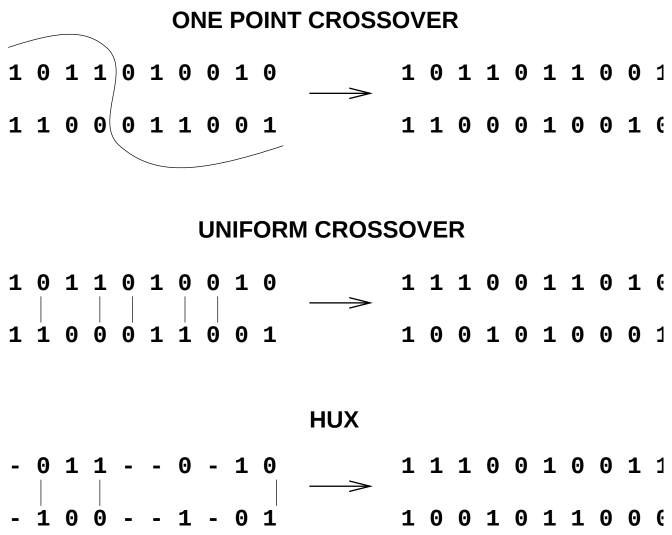
- Population-elitist selection: Truncation Selection
- Incest Prevention
- HUX
- Restarts

GENITOR MODEL



CHC MODEL

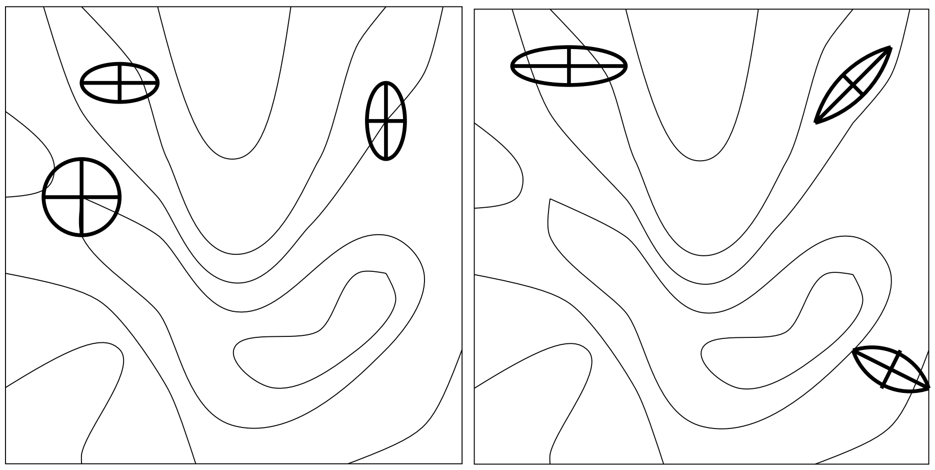




EVOLUTION STRATEGIES

- **Uses Real-Valued Parameter Representation**
- (μ, λ)-selection:
  λ Offspring replace μ Parents
- $(\mu + \lambda)$-selection:
  Truncation Selection
- Self Adaptive Mutation and Rotation
- Blending Recombination

  Note when $\lambda > \mu$ we generate extra offspring, then reduce back to μ.



Simple Mutations   Correlated Mutation via Rotation

CMA Covariance Matrix Adaptation

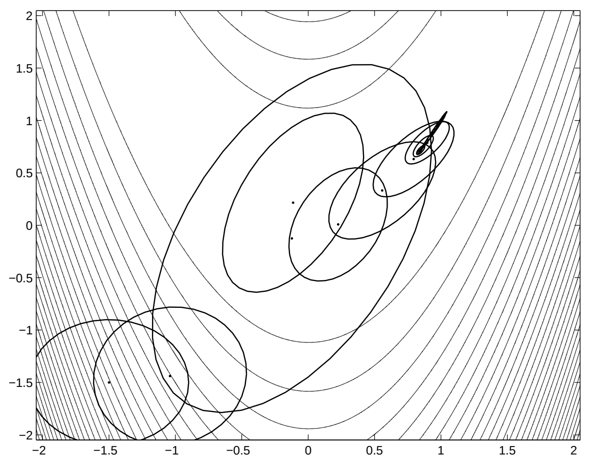
Let $\mathbf{Z}^{(g+1)}$ be the covariance of the μ best individuals.

Let $\mathbf{P}^{(g+1)}$ be the covariance of the evolution path.

The new covariance matrix is:

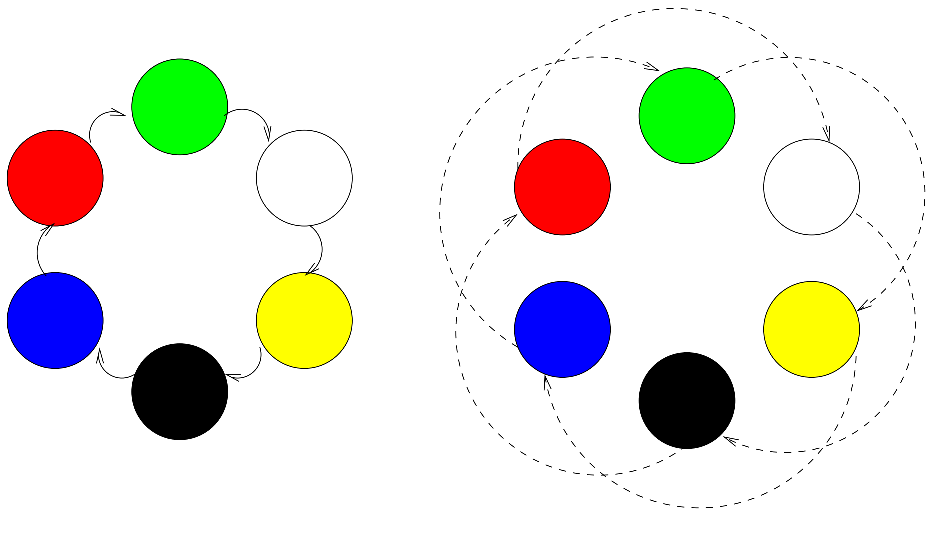
$$\mathbf{C}^{(g+1)} = (1 - c_{cov})\mathbf{C}^{(g)} + c_{cov}\left(\alpha_{cov}\mathbf{P}^{(g+1)} + (1 - \alpha_{cov})\mathbf{Z}^{(g+1)}\right)$$

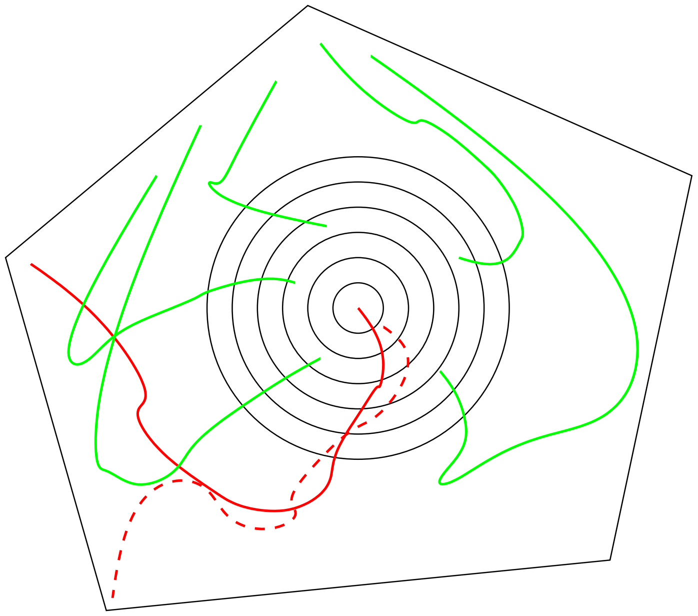
Where c_{cov} and α_{cov} are constants that weight each input.



Parallel Island Model Genetic Algorithms

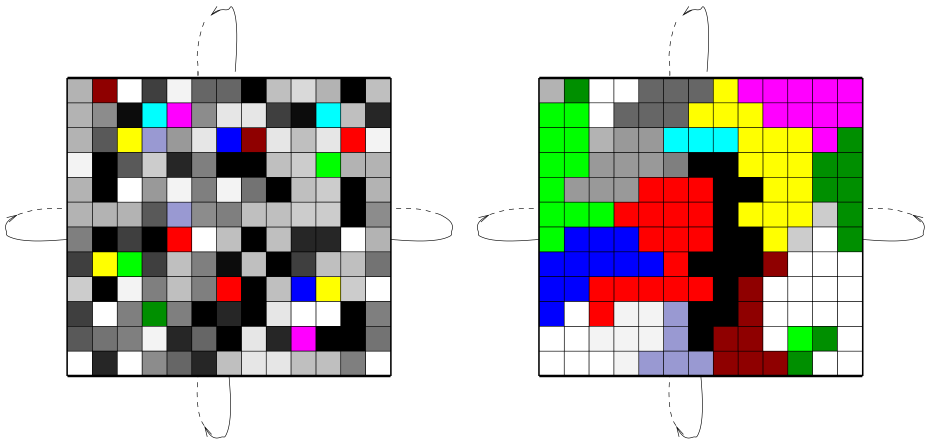
ISLAND MODEL WITH MIGRATION



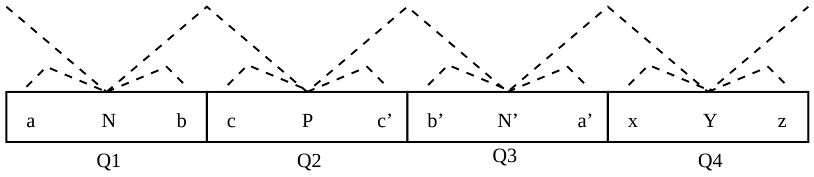


Parallel Cellular Genetic Algorithm

CELLULAR GENETIC ALGORITHM MODEL



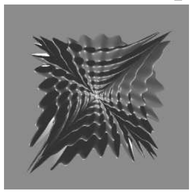
Local Quad Search



"Quad Search" uses only 4 neighbors, and evaluates only 2.
On unimodal functions it is proven to converge to optimal
in less than 2L evaluations.

A Hybrid: Genetic Quad Search

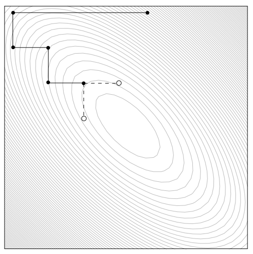
We used GENITOR, a steady-state GA, as the genetic algorithm. We used Quad Search to improve each member of the population.

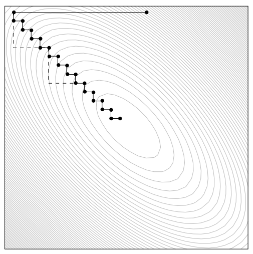


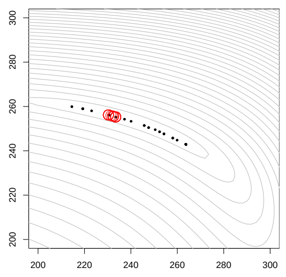
Function	Algorithm	Mean	σ	Solved	Evals
Rana 10-D	CMA-ES	-388.0	15.0	0	500K
	Quad	-434.8	8.4	0	500K
	Genitor	-443.4	17.8	0	500K
	CHC	-495.5	5.5	0	500K
	Hybrid Quad	-510.3	2.5	26	268K

Ruffled by Ridges:
How Evolutionary Algorithms Can Fail

- *Direction* – coordinate search cannot see improving points that fall between axis.
- *Precision* – increasing precision generally decreases the number of false optima.







The Temperature Inversion Problem

Researchers have created a *forward model* that relates 43 vertical temperature profiles ($\vec{x}$) to 2,000 observed measurements ($\vec{y}$).

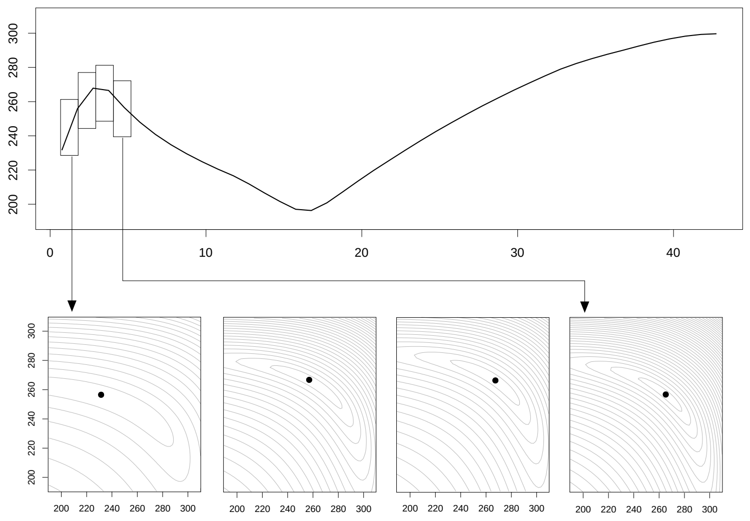
- model($\vec{x}$) $\longrightarrow \vec{y}$
- An analytical inversion of this model is impossible.
- Formulate as an optimization problem:

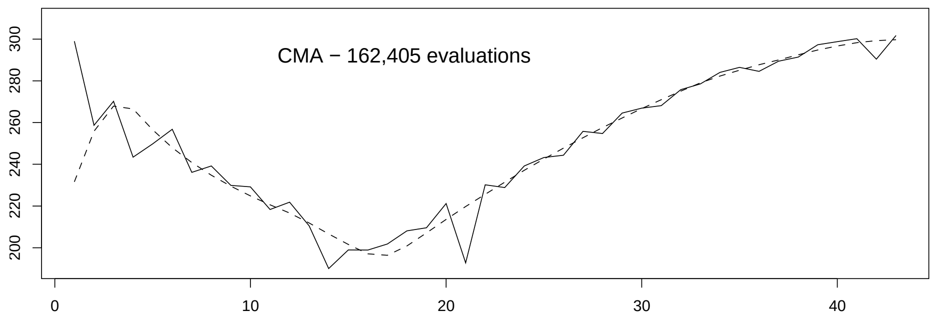
$$f(\vec{x}) = (\vec{y}_{obs} - \text{model}(\vec{x}))^T(\vec{y}_{obs} - \text{model}(\vec{x}))$$

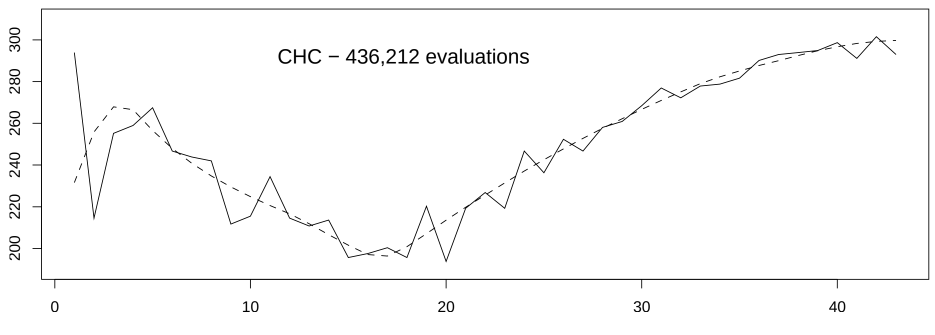
- Sometimes first order derivatives can be calculated analytically.
  In the general case, this is impossible.

Why is the temperature problem so hard?

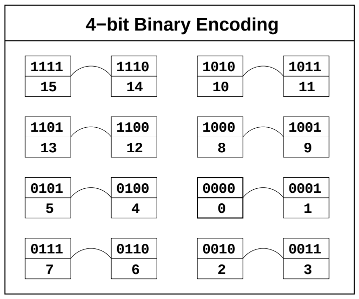
- Ridges in search space.

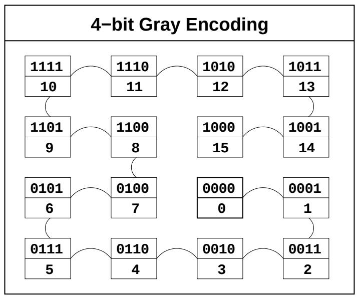


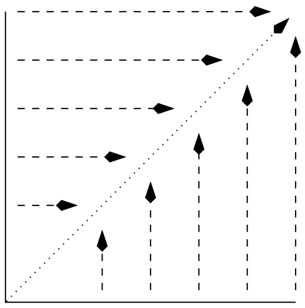




Gray vs Binary vs Real







There are good arguments for Gray codes over Binary.
But Gray codes are “blind” to ridges.
Sometimes Binary is better.

Gray vs Binary vs Real

Comparing “Real-Valued” and “Bit” representation is much more complex than most of the literature suggests.

Genetic algorithms at 20 bits of precision can be 10 to 100 times slower to converge using 20 versus 10 bits of precision.

Low Precision might ”miss” good solutions.
But it aids exploration.

High Precision can result in low/slow exploration.

The Testing Problem

1. Test Functions can be TOO EASY
   E.G. ONEMAX, Sphere Functions
2. Test Functions can be TOO HARD
   E.G. Random Job Shop Problems, N-K Landscapes
3. Test Functions can be UNREALISTIC
   E.G. Deceptive Functions and Trap Functions
   (These have theoretical value, but ....)
4. Test Functions can be TOO SPECIALIZED
   E.G. MAXSAT: Too many flat plateaus

There are no easy answers.

Parameter Optimization and Test Problems

There are many common test functions. Not all are good test functions. Many are linearly separable:

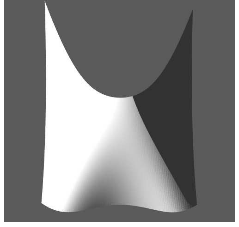
$$F(x, y, z) = G(x) + G(y) + G(z)$$

Pairwise combinations can be used to introduce greater nonlinearity:

$$F(x, y, z) = G(x, y) + G(y, z) + G(z, x)$$

where $G(x, y)$ is a nonlinear function of x and y.

Test functions can also be **rotated** *to create nonlinearity.*



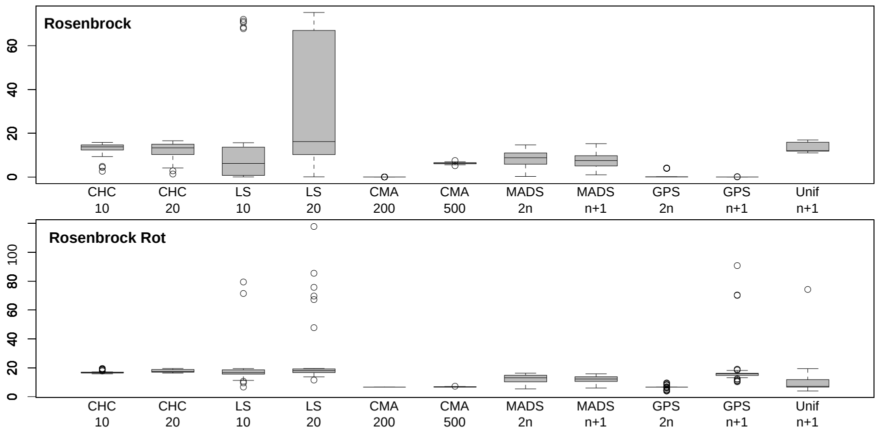
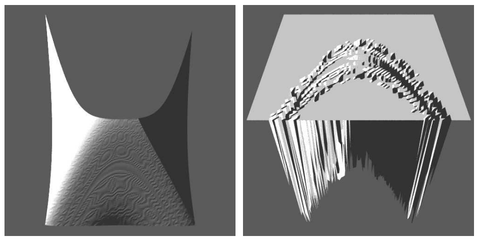


Figure 1: The midline=median; the gray box represents 50 percent of evaluations. The bars show max and min values, except for outliers (small circles).



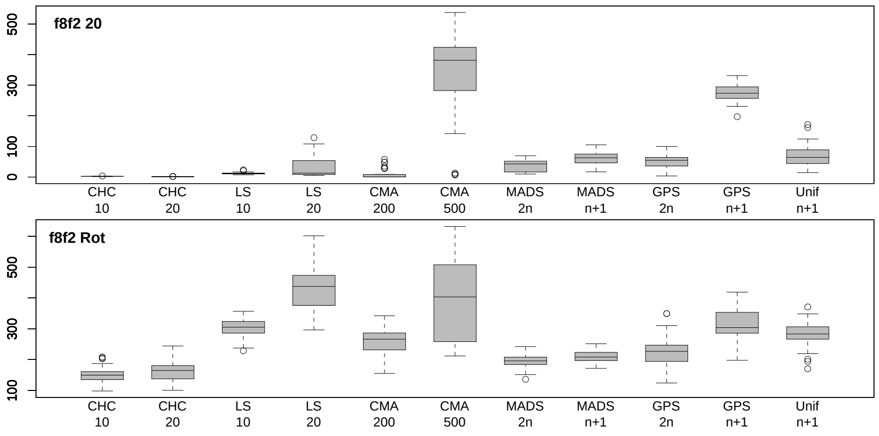
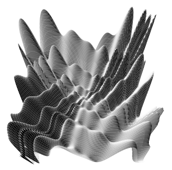


Figure 2: The midline=median; the gray box represents 50 percent of evaluations. The bars show max and min values, except for outliers (small circles).



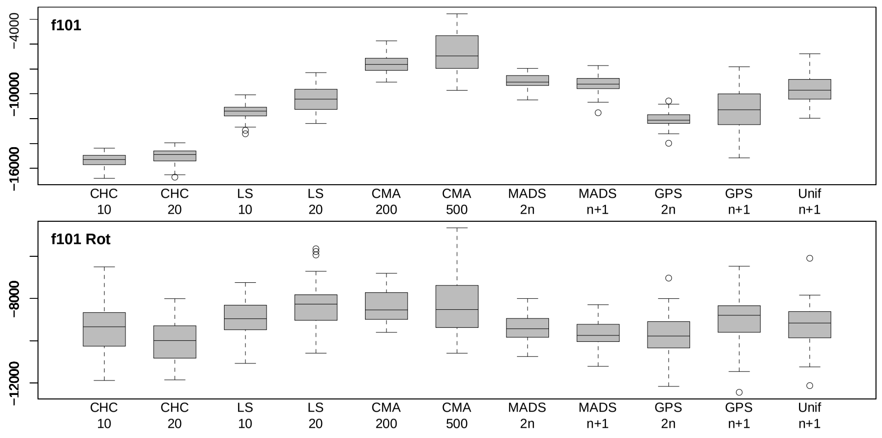
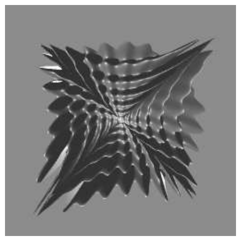


Figure 3: The midline=median; the gray box represents 50 percent of evaluations. The bars show max and min values, except for outliers (small circles).



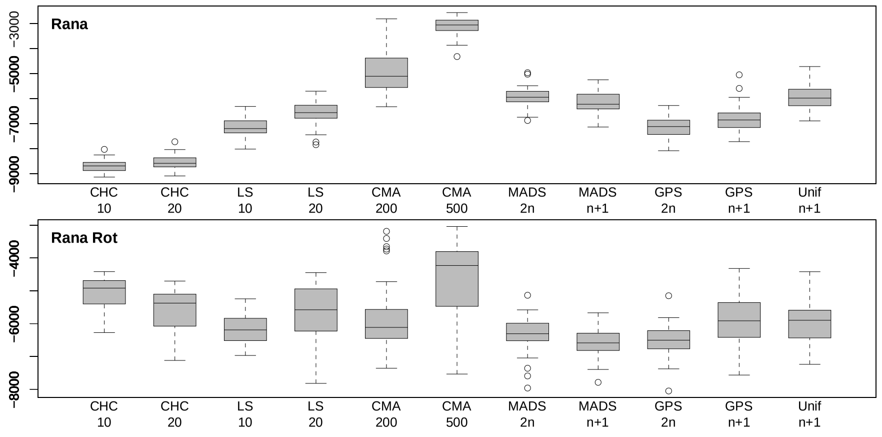


Figure 4: The midline=median; the gray box represents 50 percent of evaluations. The bars show max and min values, except for outliers (small circles).

NFL: No Free Lunch

All search algorithms are equivalent when compared over all possible discrete functions. Wolpert, Macready (1995)

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. Resampling is ignored. 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” versus a “WorstFirst” local search with restarts. For every j there exists an l such that

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

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:  A B C          F1:  0 0 0 1
F2:  A C B          F2:  0 0 1 0
F3:  B A C          F3:  0 1 0 0
F4:  B C A          F4:  1 0 0 0
F5:  C A B
F6:  C B A
```

POSSIBLE ALGORITHMS		POSSIBLE 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

QUESTION:

How should we evaluate search algorithms?

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

If algorithm **S** *is better than algorithm* **T** *on* β*...*
Then **T** *is better than* **S** *on* $P(\beta) - \beta$*.*
This is True in the aggregate, but not on average.

EXPERIMENT

Let's evolve Lisp Programs ala Genetic Programming.
But let's try different evolutionary algorithms.
The GAs use mutation and tree-based crossover. The ESs use mutation only.

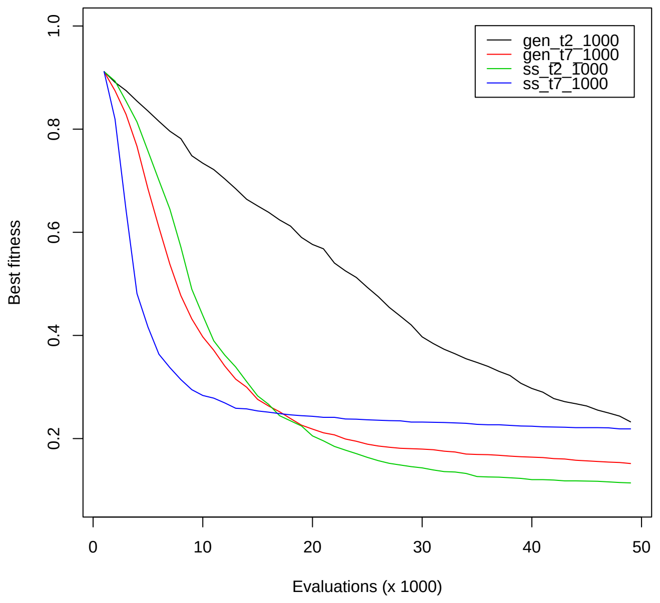
We will evolve

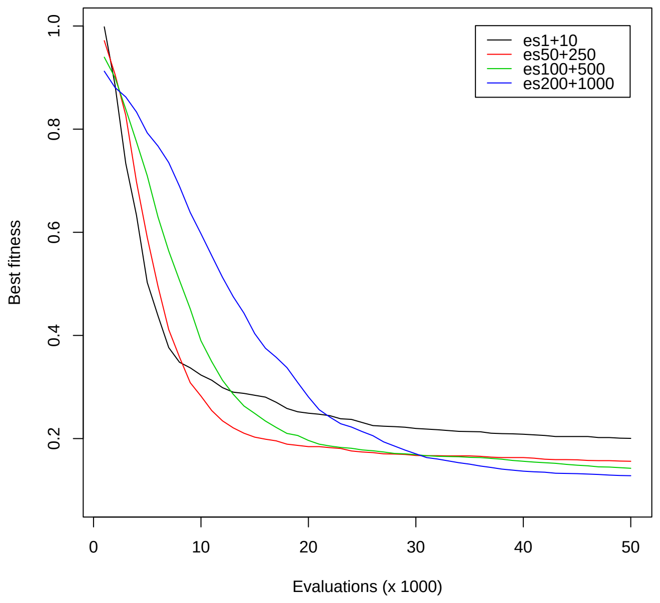
- **11-Multiplexer**
- **Pole Balancing with Cart Centering**
- **Symbolic Regression**

We will compare

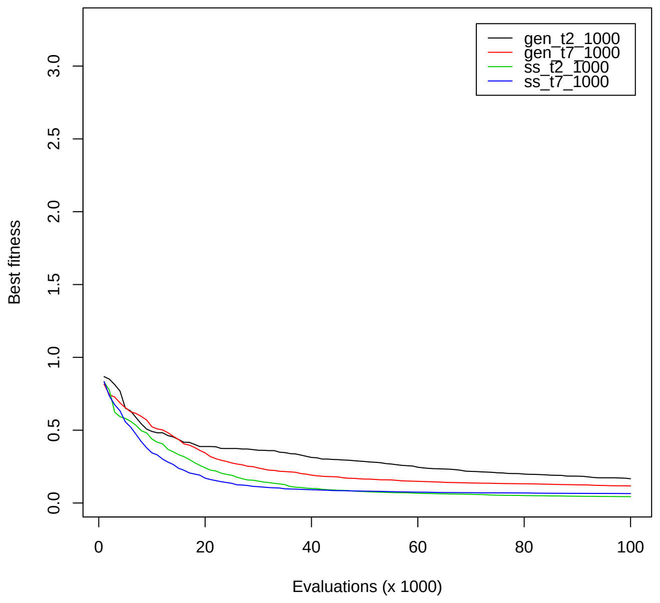
- A Generational GA, Popsize = 1000, Tournament Size 2 and 7
- A Steady-State GA, Popsize = 1000, Tournament Size 2 and 7
- Evolution Strategies: (1+1)ES, (50+250)ES, (100,500)ES, (200,1000)ES

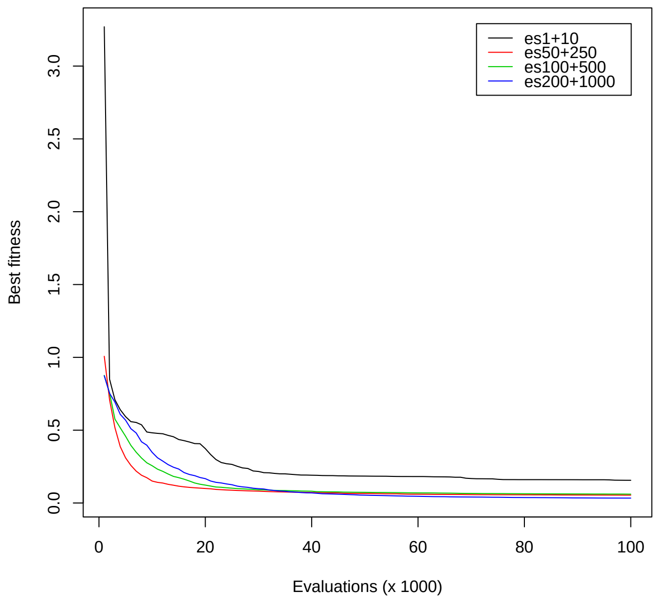
Pole Balancing



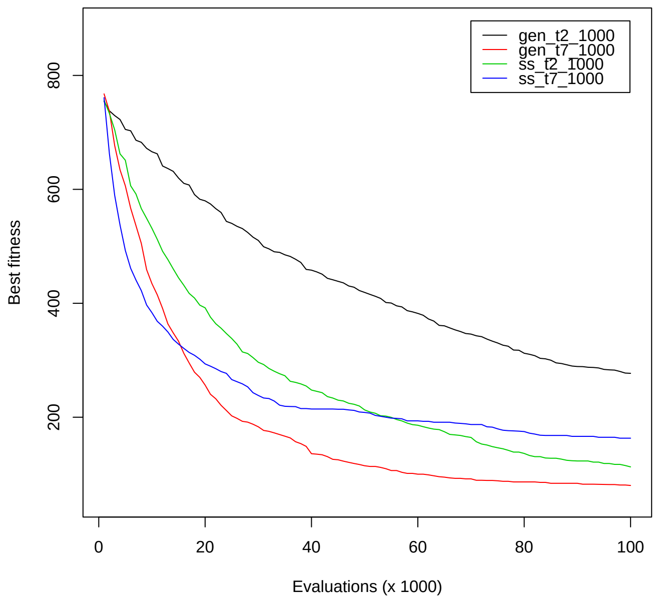


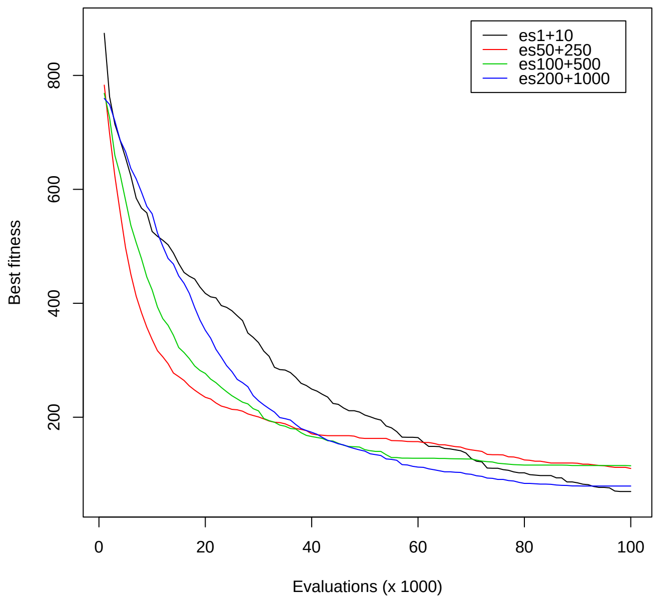
Symbolic Regression: $x^6 - 2x^4 + x^2$





The 11 Multiplexer


gen_t2_1000
gen_t7_1000
ss_t2_1000
ss_t7_1000
Best fitness
Evaluations (x 1000)


es1+10
es50+250
es100+500
es200+1000
Best fitness
Evaluations (x 1000)