Evolutionary Algorithms for Optimization

Darrell Whitley
Colorado State AI Lab
Colorado State University

OVERVIEW

EVOLUTION STATEGIES

GENETIC ALGORITHMS

CMA–EVOLUTION STRATEGIES

EVOLUTIONARY PROGRAMMING

CHC

MEMETIC ALGORITHMS

HYBRID EVOLUTIONARY ALGORITHMS

PARALLEL ISLAND MODELS

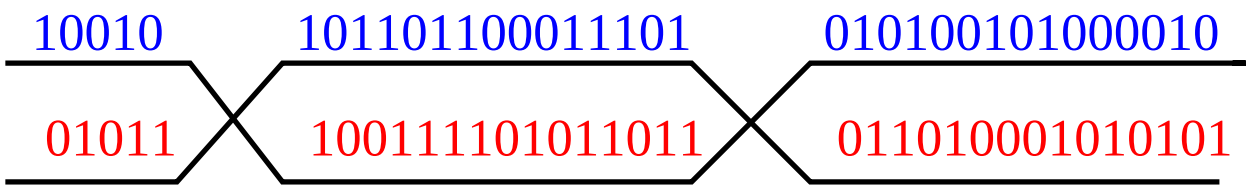
Local Search, Tabu Search, Generalized Pattern Search, Nelder-Mead ...

GENETIC RECOMBINATION

Let the following two binary strings represent an encoding of 5 parameters that are used in some optimization problems.

1001010 1101100 0111010 1010010 1000010

0101110 0111101 0110110 1101000 1010101

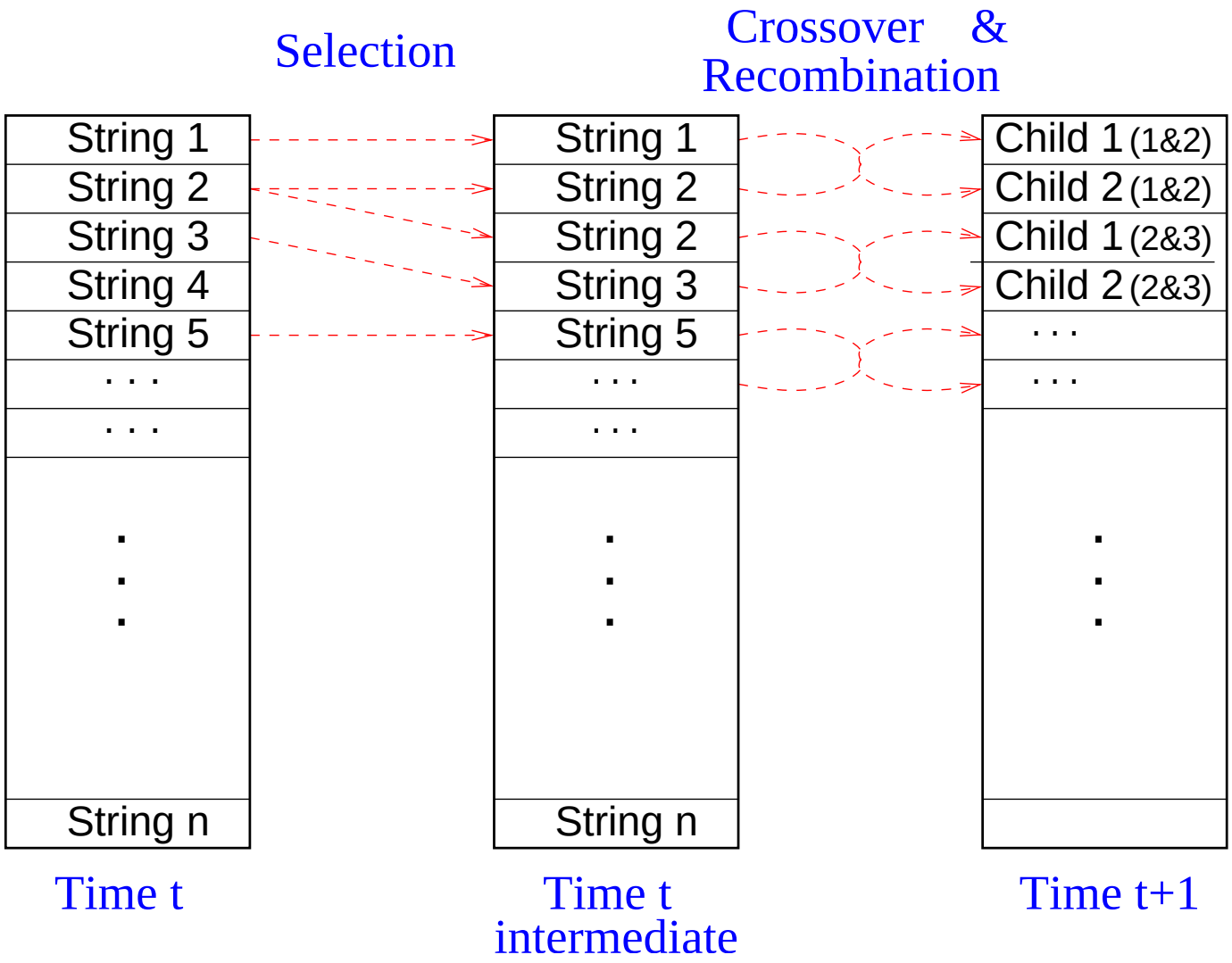


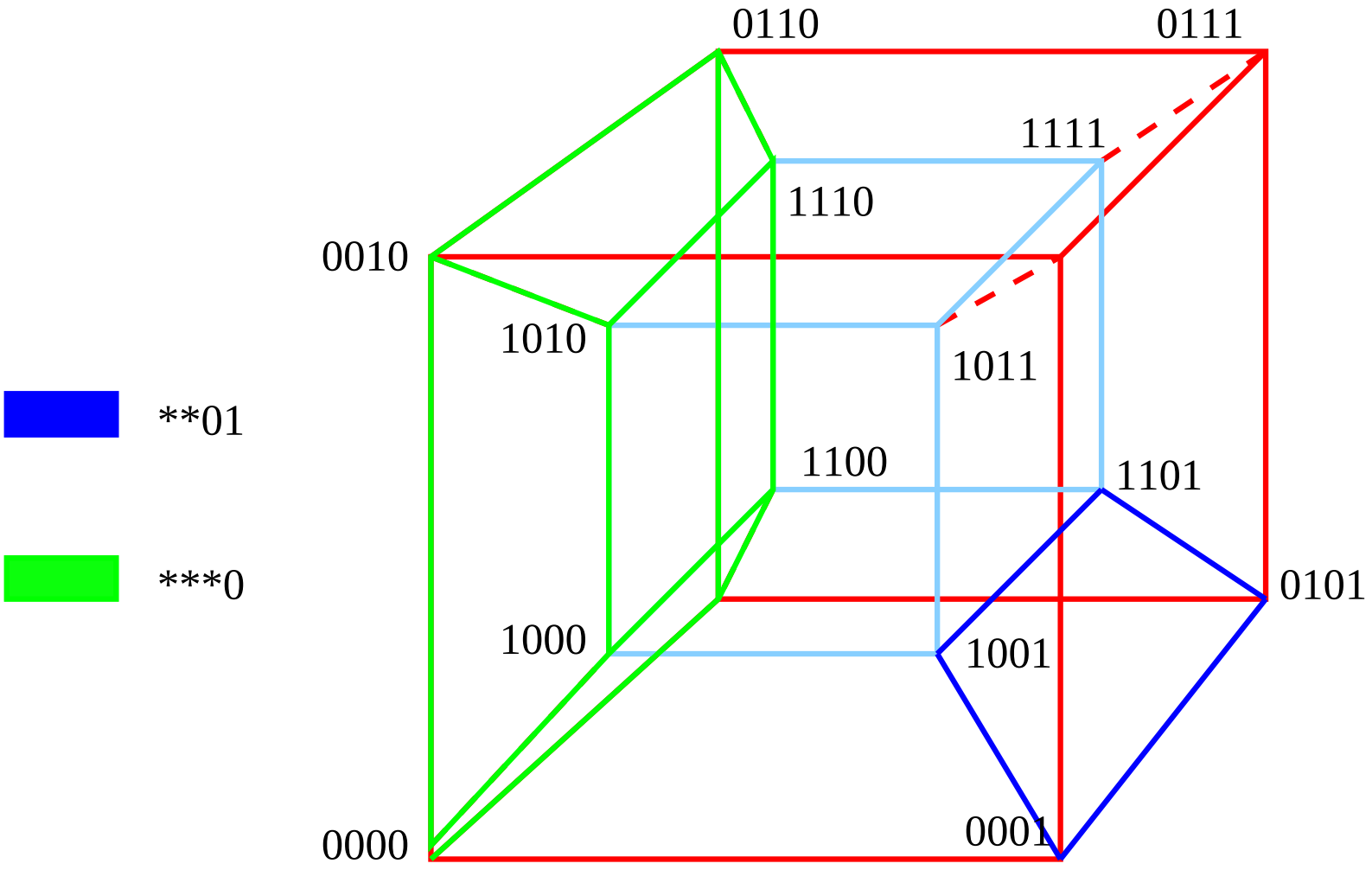
Which Produces the Offspring

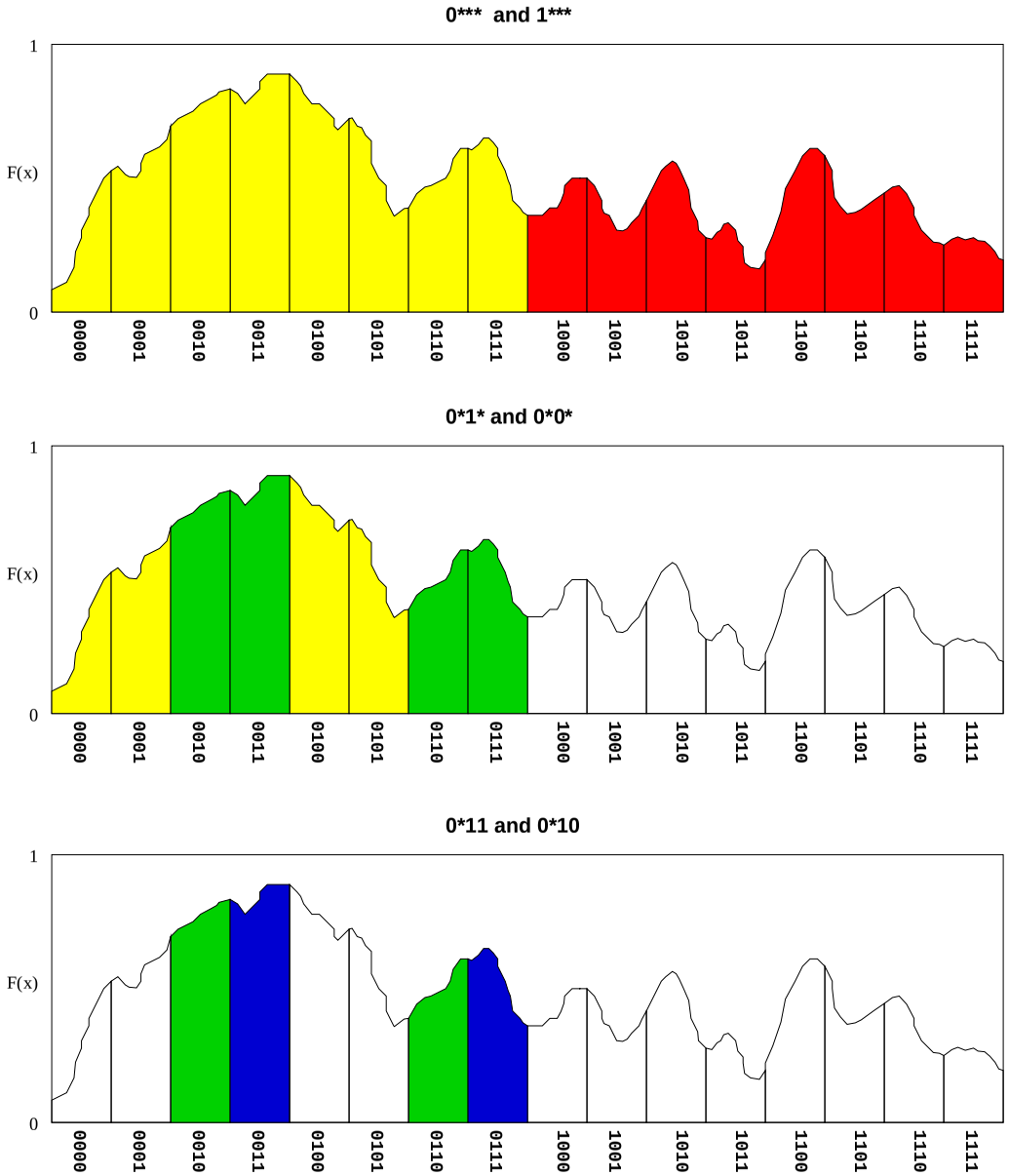
0101110110110001110101101000101010 1

1001010011110101101101010010100001 0

SIMPLE GENETIC ALGORITHM MODEL




**01
***0
0110
0111
1111
1110
0010
1010
1011
1100
1101
0101
1000
1001
0000
0001


0*** and 1***
1
F(x)
0
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
0*1* and 0*0*
1
F(x)
0
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
0*11 and 0*10
1
F(x)
0
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

THE SCHEMA THEOREM

Selection Only: $P(H, t + intermediate) = P(H, t)\frac{f(H)}{\bar{f}}$.

An Exact Calculation:

$$P(H, t+1) = (1 - p_c)P(H, t)\tfrac{f(H)}{\bar{f}} + p_c \left[P(H, t)\tfrac{f(H)}{\bar{f}}(1 - losses) + gains\right]$$

$$P(H, t+1) = P(H, t)\frac{f(H)}{\bar{f}}(1 - p_c\, losses) + p_c gains$$

A Common Version of the "Schema Theorem":

$$P(H, t+1) \geq P(H, t)\ \tfrac{f(H)}{\bar{f}}\ \left[1 - p_c \tfrac{\Delta(H)}{L-1}(1 - P(H, t)\ \tfrac{f(H)}{\bar{f}})\right](1 - p_m)^{o(H)}$$

The Vose and Liepins Model

The i th component of vector s^t is the probability that the string i is selected for the gene pool.

$$s_i^t = P(i,t)f(i)/\bar{f}$$

Construct a mixing matrix M where the i,jth entry $m_{i,j} = r_{i,j}(0)$. This matrix gives the probabilities that crossing strings i and j will produce the string $i = 0$. Then the proportional representation for string 0 at time t+1 is given by:

$$p_0 = s^T M s$$

Matrix F stores the fitness values along the diagonal.

$$s^{t+1} = \frac{Fp^{t+1}}{1^T Fp^{t+1}}$$

MUTATION

Let M_1 be the recombination matrix.

Define $\mathcal{Q}$ as the mutation matrix.

Mutation can be done after crossover: $p^T \mathcal{Q}$

Mutation can be done before crossover: $s^T \mathcal{Q}$ or $\mathcal{Q}^T s$

$$p_0{}^{t+1} = (\mathcal{Q}^T s)^T M_1 (\mathcal{Q}^T s)$$

$$p_0{}^{t+1} = s^T (\mathcal{Q} M_1 \mathcal{Q}^T) s$$

$$p_0{}^{t+1} = s^T (\mathcal{Q} M_1 \mathcal{Q}^T) s$$

$$p_0{}^{t+1} = s^T M s \quad \text{where} \quad M = (\mathcal{Q} M_1 \mathcal{Q}^T)$$

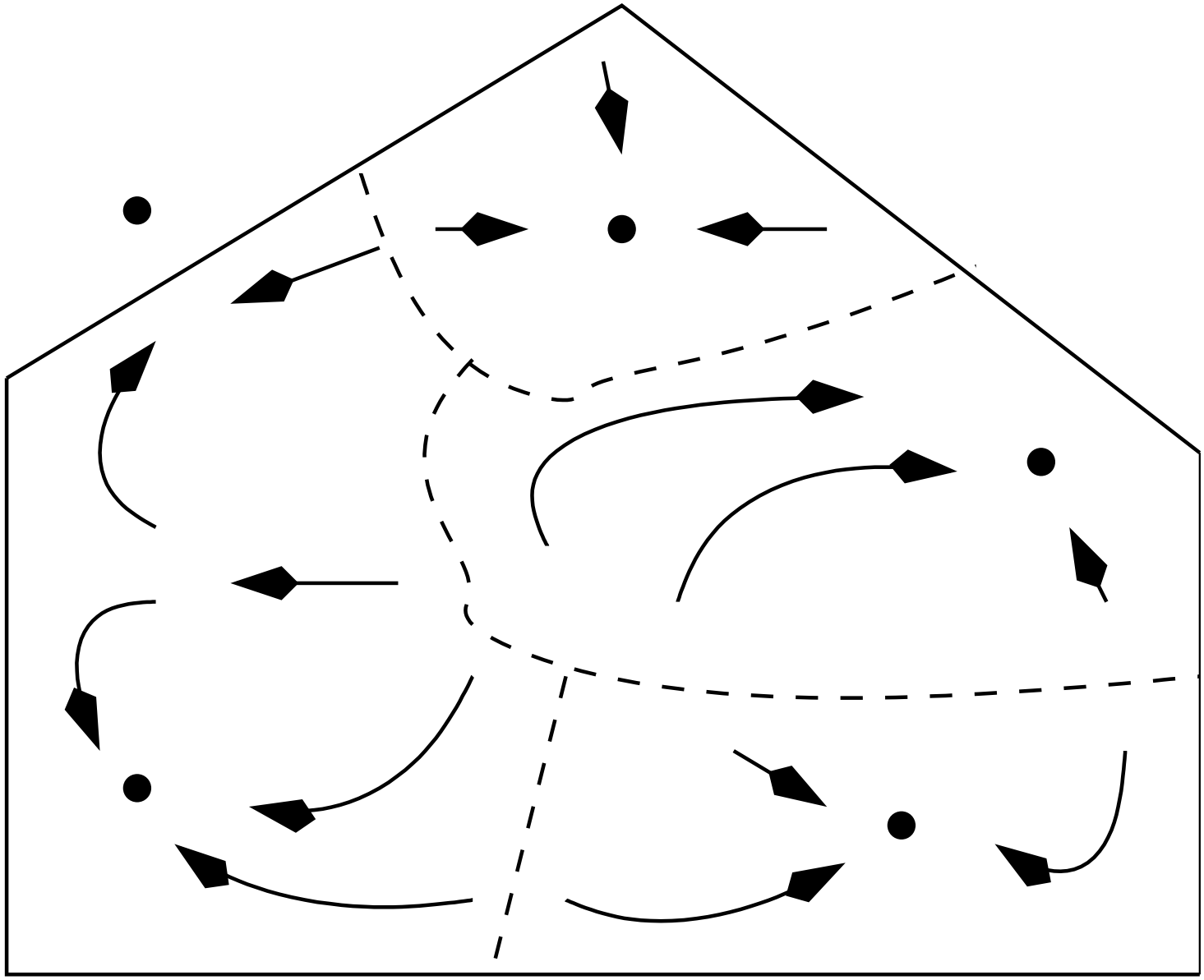
A Transform Function using bit-wise exclusive-or: $\oplus$

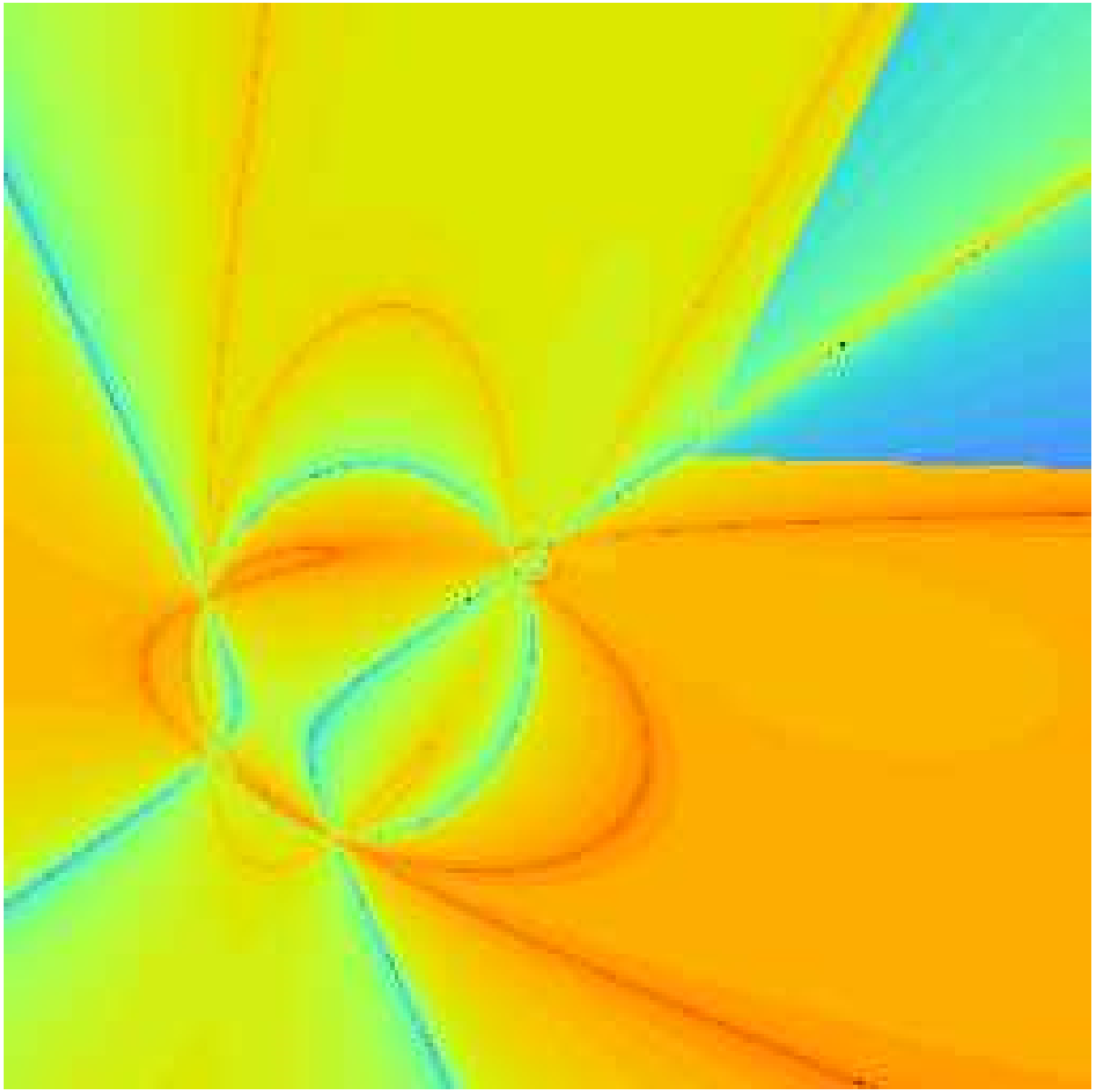
$$
\begin{aligned}
000 \oplus 010 &\Rightarrow 010 \\
001 \oplus 010 &\Rightarrow 011 \\
010 \oplus 010 &\Rightarrow 000 \\
011 \oplus 010 &\Rightarrow 001 \\
100 \oplus 010 &\Rightarrow 110 \\
101 \oplus 010 &\Rightarrow 111 \\
110 \oplus 010 &\Rightarrow 100 \\
111 \oplus 010 &\Rightarrow 101
\end{aligned}
$$

Let $r_{i,j}(k)$ be the probability that k results from the recombination of strings i and j. If recombination is a combination of crossover and mutation then

$$r_{i,j}(k \oplus q) = r_{i \oplus k, j \oplus k}(q) \text{ which implies } r_{i,j}(k) = r_{i \oplus k, j \oplus k}(0).$$

We use this to construct $G(p^t)$ which is the exact trajectory of an infinite population: $p^{t+1} = G(p^t)$





MARKOV MODELS

The Markov Model is an N X N transition matrix Q, where $Q_{i,j}$ is the probability that the k^{th} generation is population $\mathcal{P}_j$ given that the $(k-1)^{th}$ population is $\mathcal{P}_i$.

Let $< Z_0, Z_1, Z_2, ..., Z_r - 1 >$ represent a population, where Z_k represents the number of copies of string k in population and $r = 2^L$.

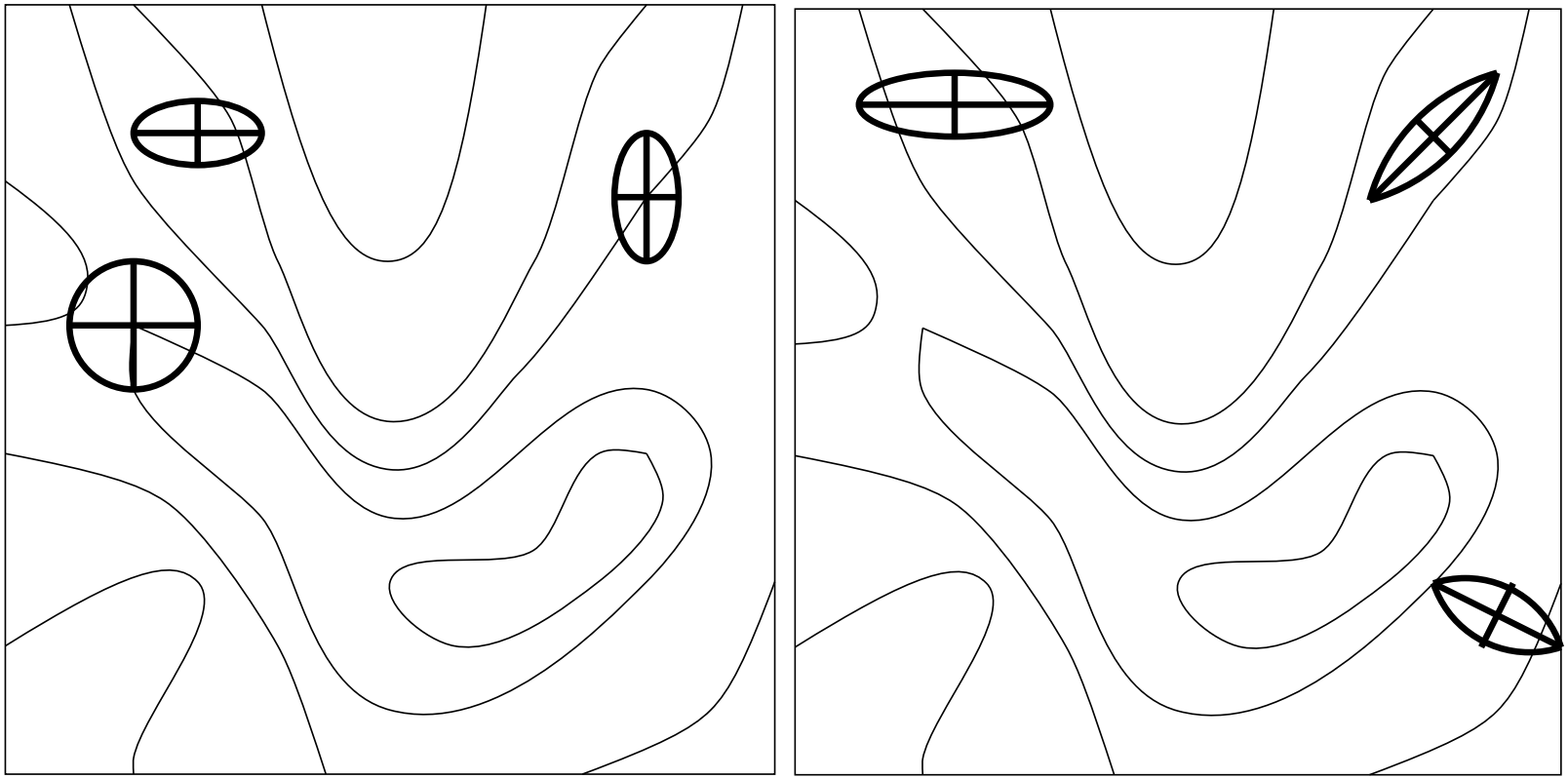
Vector p vector represents the distribution of an infinite population, and the probability distribution for generating any single string.

$$Q_{i,j} = K! \prod_{y=0}^{r-1} \frac{(G(p^t)_y)^{Z_y}}{Z_y!}$$

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

$N(0,1)$ normally distributed 1-D random variable, zero mean $\sigma = 1.0$.
$N_i(0,1)$ the same function, a new sample for each i.
τ, τ' and β denote constants that control step sizes.

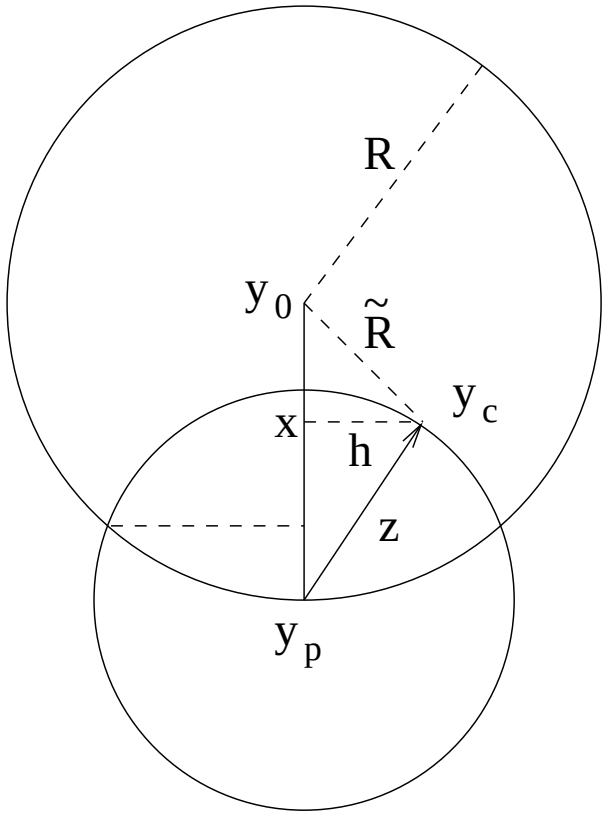
Mutation acts on a chromosome $< \vec{x}, \vec{\sigma}, \vec{\alpha} >$
to creat a new chomosome $< \vec{x}', \vec{\sigma}', \vec{\alpha}' >$

The new step size: $\sigma_i' = \sigma_i \exp(\tau' N(0,1) + \tau N_i(0,1))$

The new rotations: $\alpha_j' = \alpha_j + \beta N_j(0,1)$

The new object parameters: $\vec{x}' = \vec{x} + \vec{N}(\vec{0}, \mathbf{C}(\vec{\sigma}', \vec{\alpha}'))$

where $\mathbf{C^{-1}}$ is a covariance matrix constructed from $\vec{\sigma}'$ and $\vec{\alpha}'$.



The 1/5 rule for the Sphere Function.

When the step size is adapted so that 1 mutation in 5 is an improving move, the speed to the optimum is (approximately) maximized.

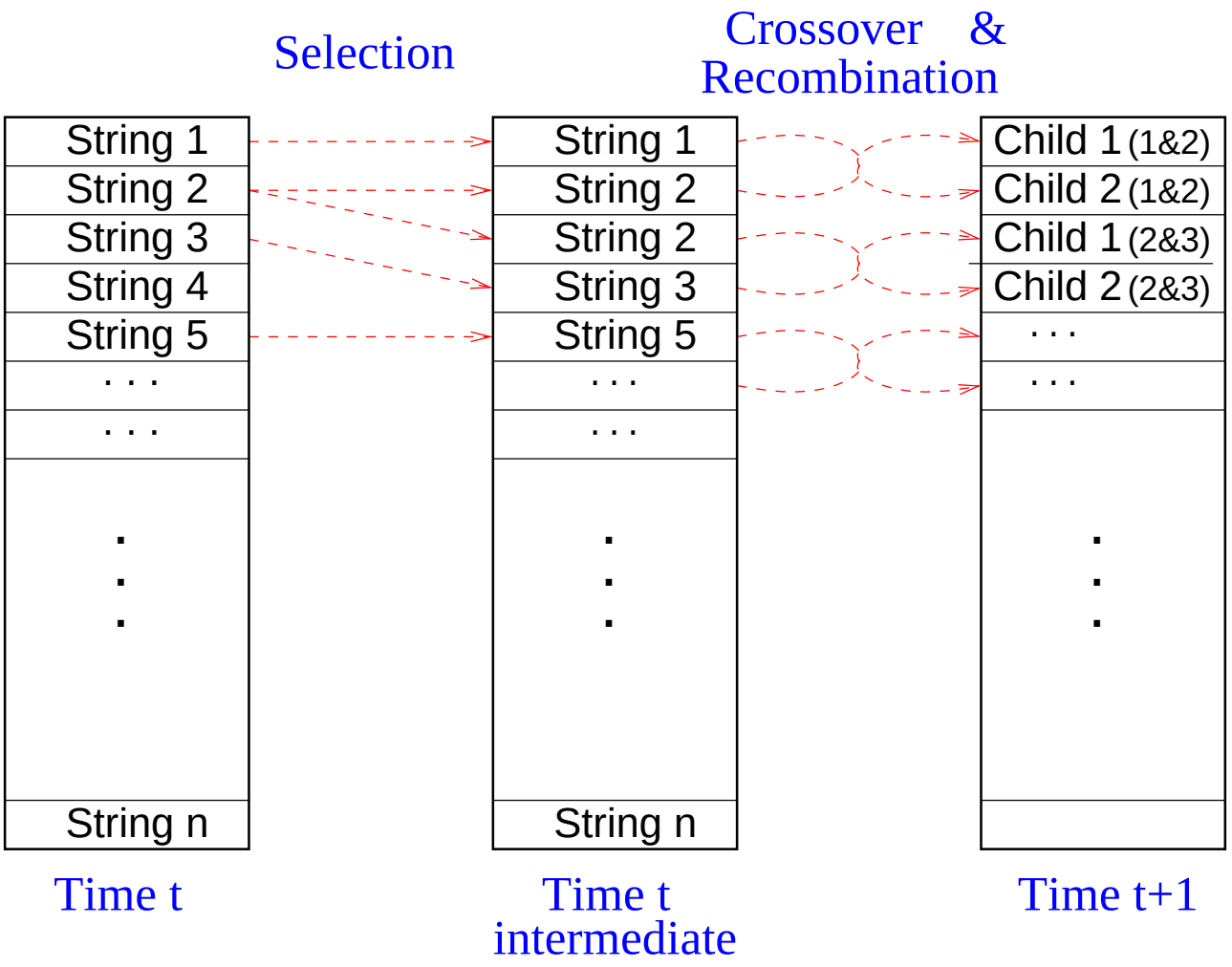
A Sample Set of Evolutionary Algorithms

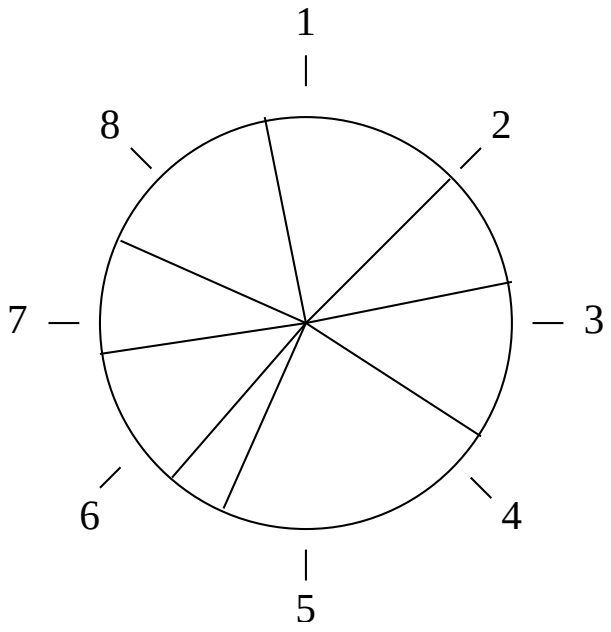
- **Simple Genetic Algorithm**: Holland/Goldberg
- **Evolution Strategies**: Schwefel/Rechenburg
- **Genitor, Steady-State GAs**: Whitley
- **CHC**: Eshelman
- **CMA Evolution Strategies**:Hansen, Ostermeier
- **Parallel Genetic Algorithms**
  - **Island Model Genetic Algorithms**
  - **Cellular Genetic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

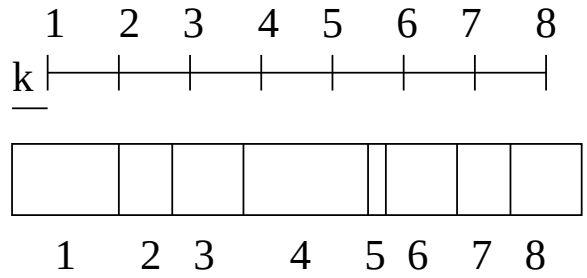
The Simple Genetic Algorithm (with Elitism)

- Roulette Wheel Selection
- One Point Crossover
- Mutation
- *Elitism*

SIMPLE GENETIC ALGORITHM MODEL

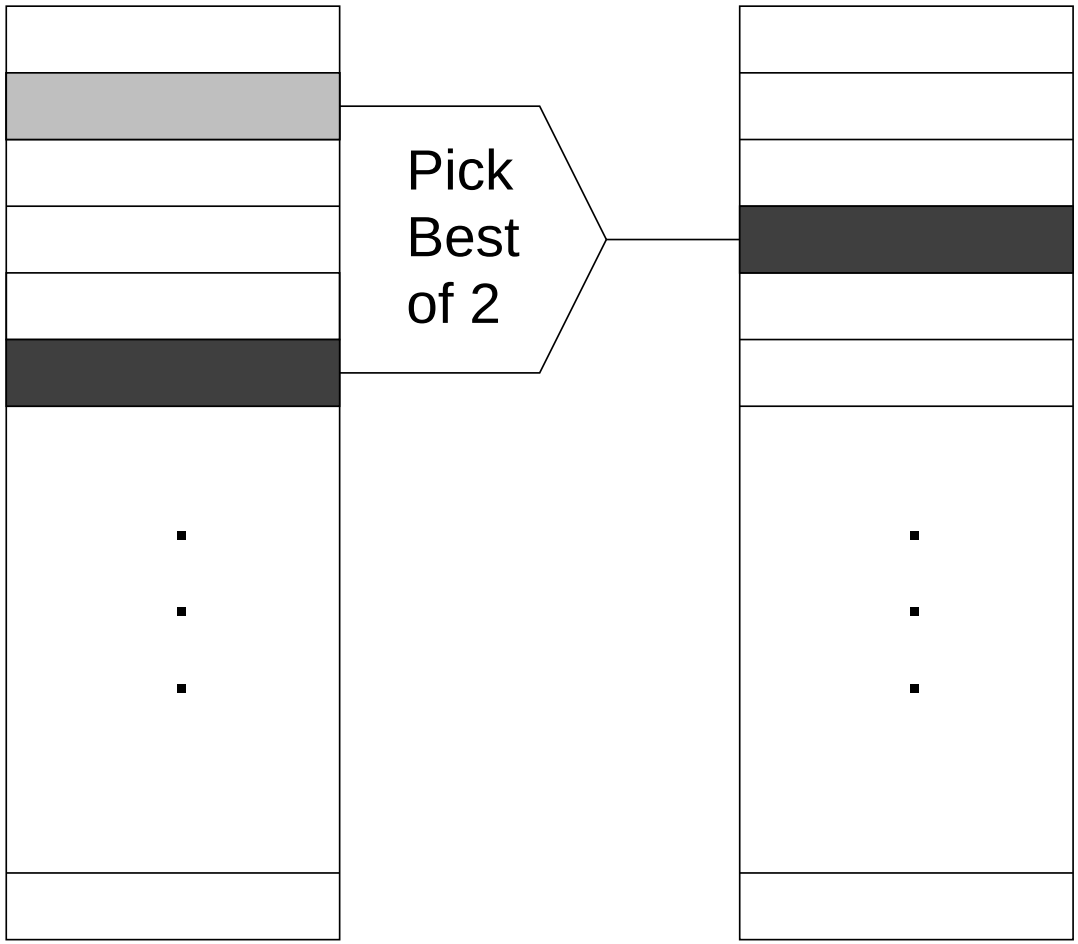






Universal Stochastic Sampling, Roulette Wheel Selection

TOURNAMENT SELECTION



A Less Noisy Form of Tournament Selection

Assume a population of size K

For $i = 1$ to K

Compare the i^{th} member of the population

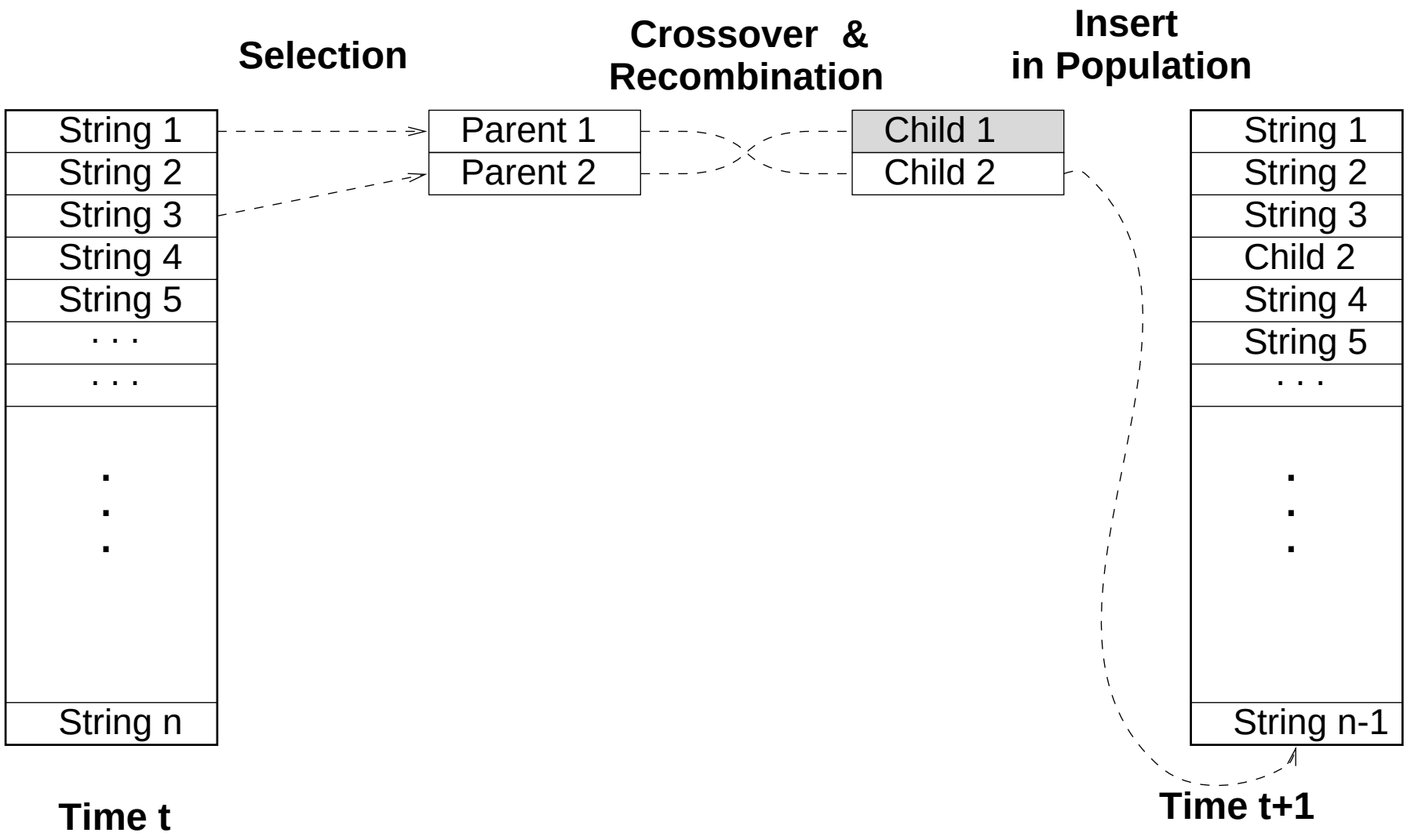
against a random member of the population.

Keep the best.

Genitor: A "Steady State" GA

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

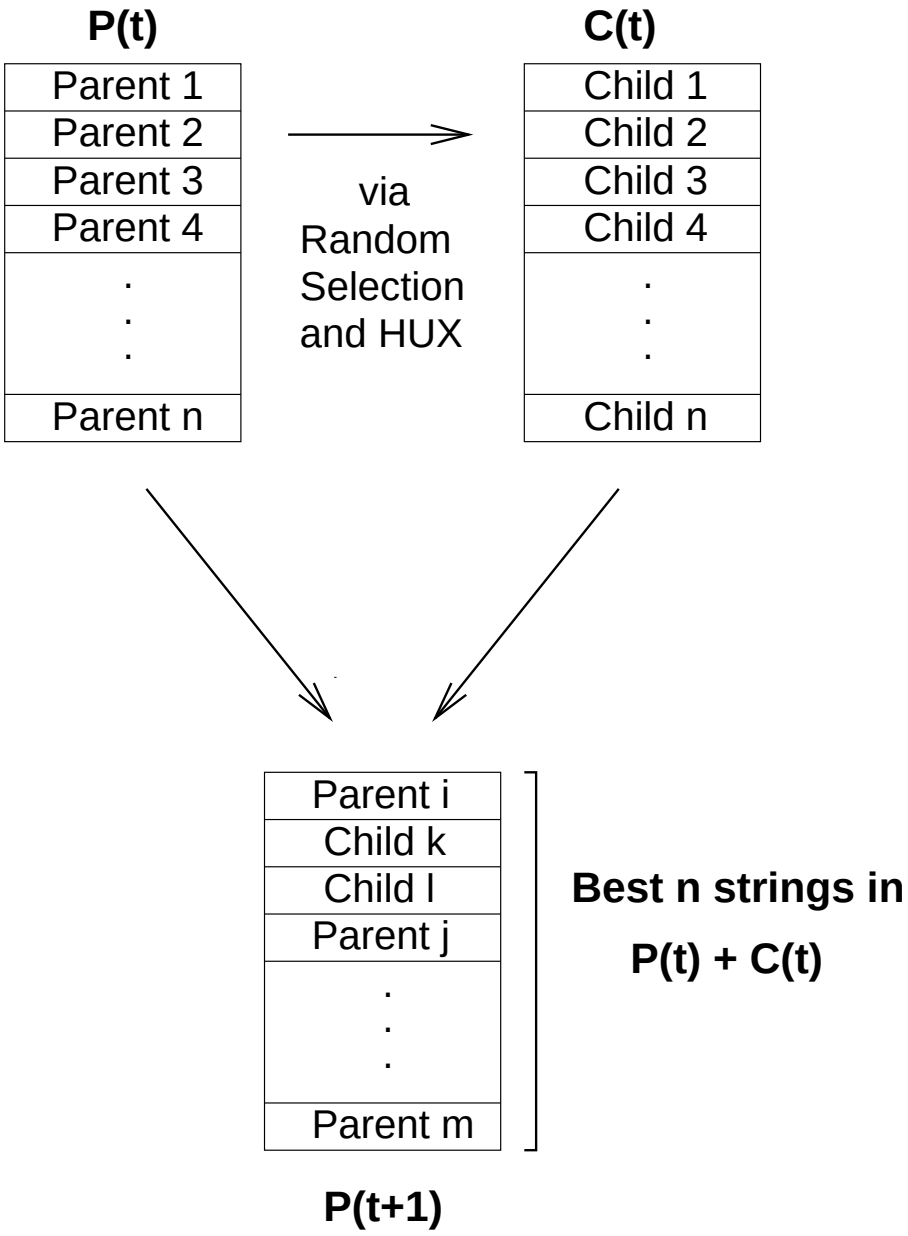
GENITOR MODEL



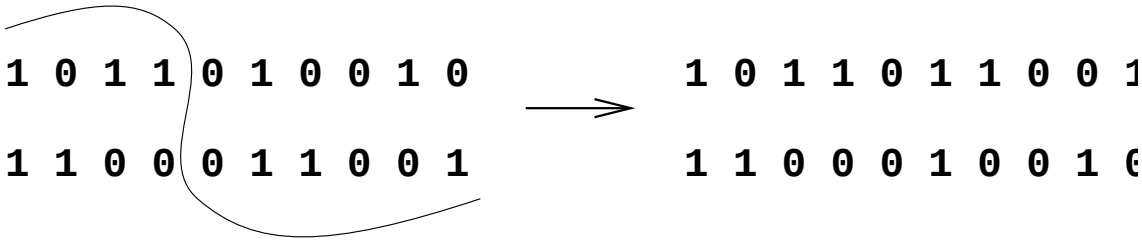
CHC

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

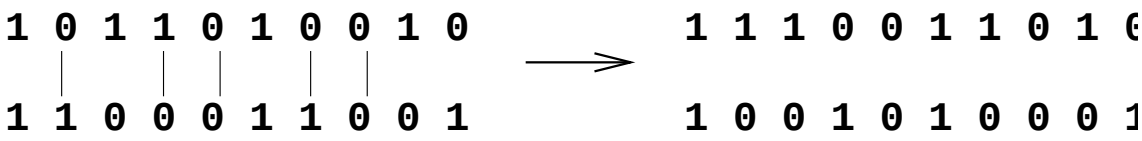
CHC MODEL


P(t)
Parent 1
Parent 2
Parent 3
Parent 4
Parent n
via
Random
Selection
and HUX
C(t)
Child 1
Child 2
Child 3
Child 4
Child n
Parent i
Child k
Child l
Parent j
Parent m
Best n strings in
P(t) + C(t)
P(t+1)

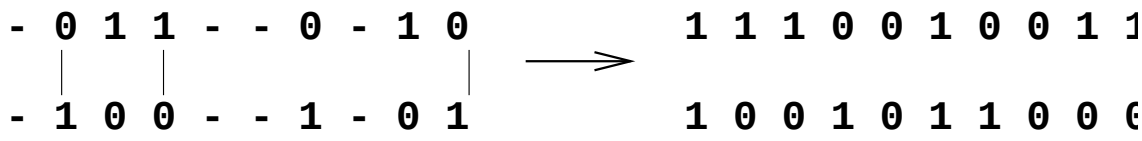
ONE POINT CROSSOVER



UNIFORM CROSSOVER



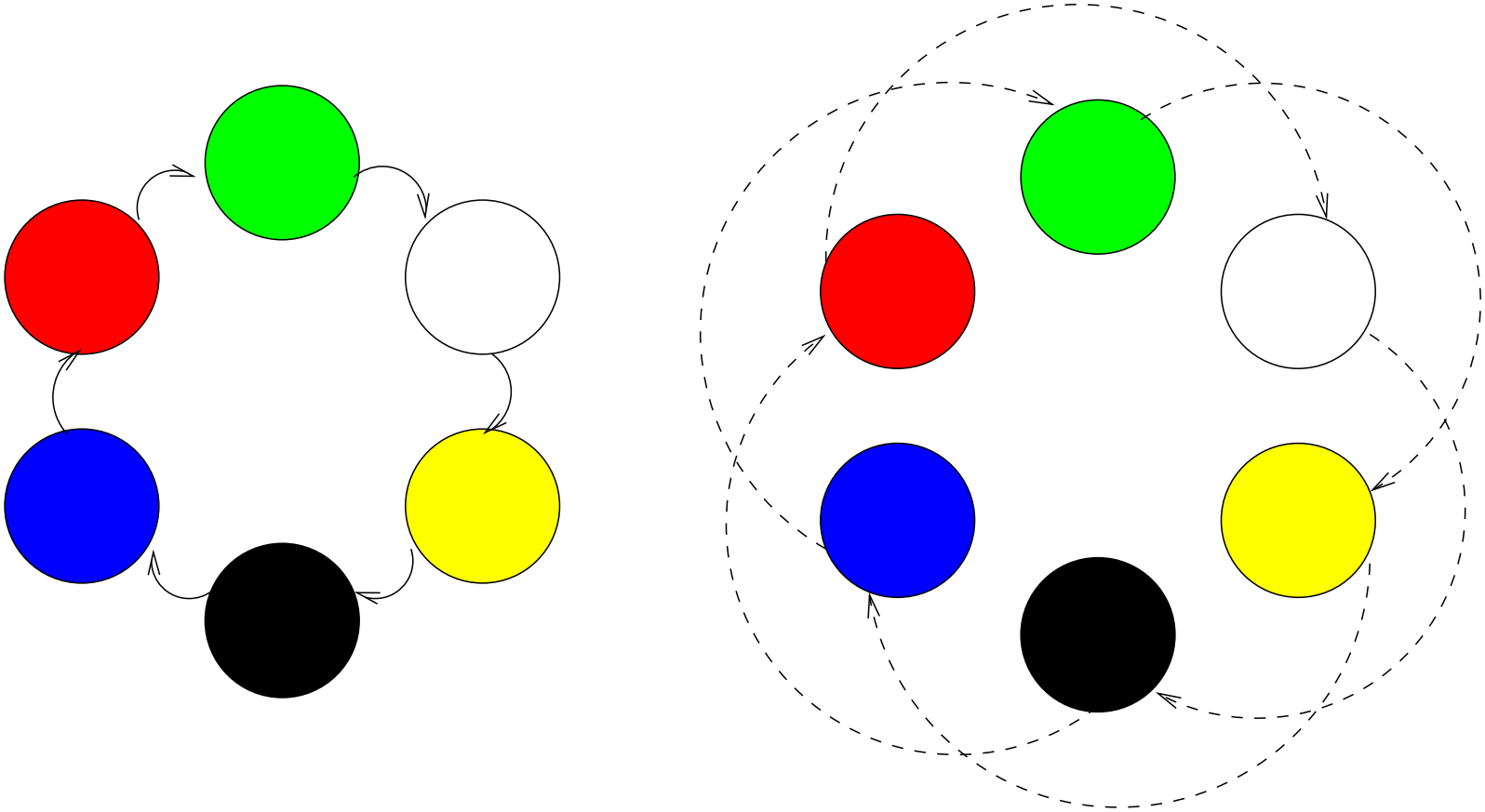
HUX

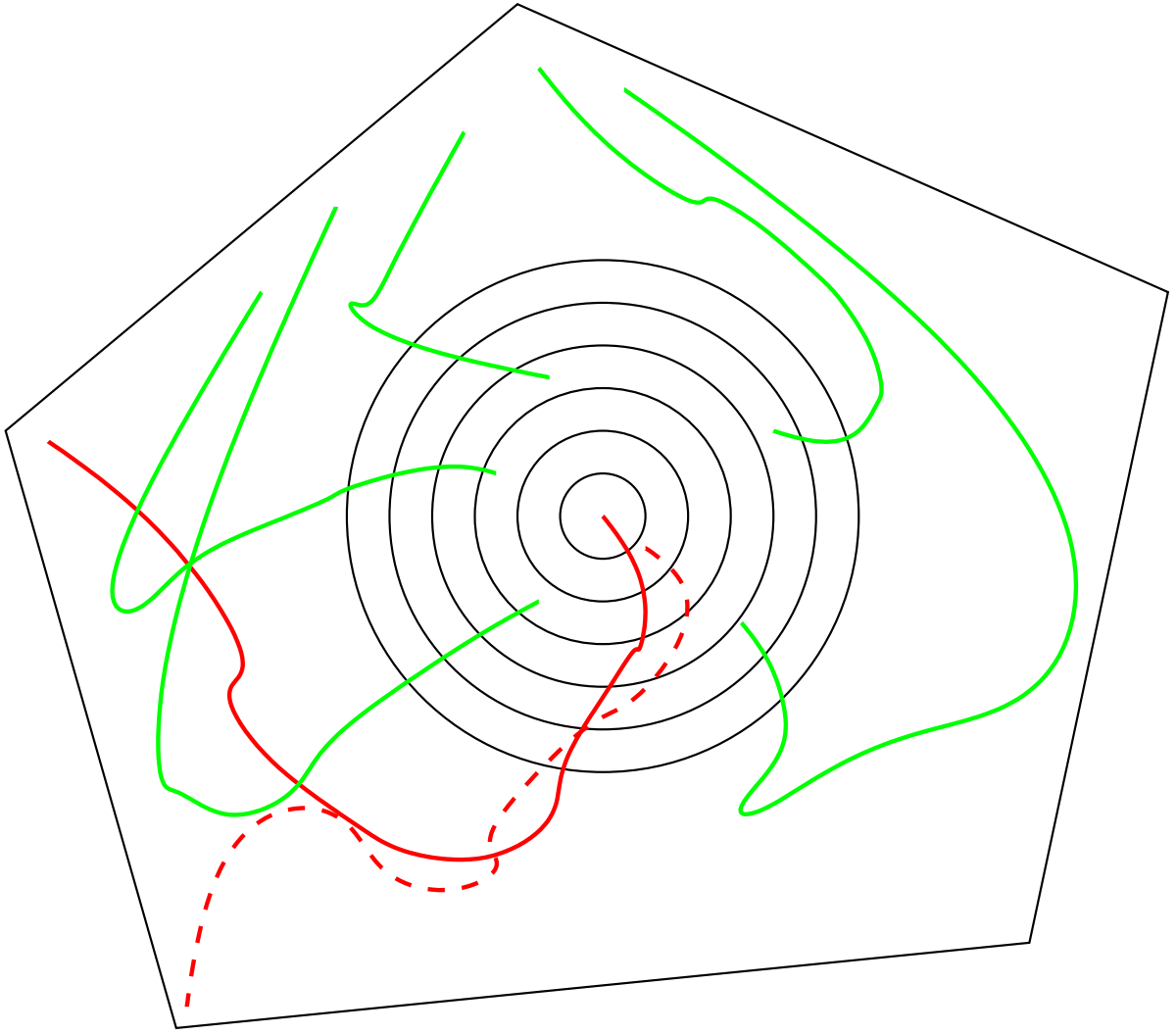


Parallel Genetic Algorithms

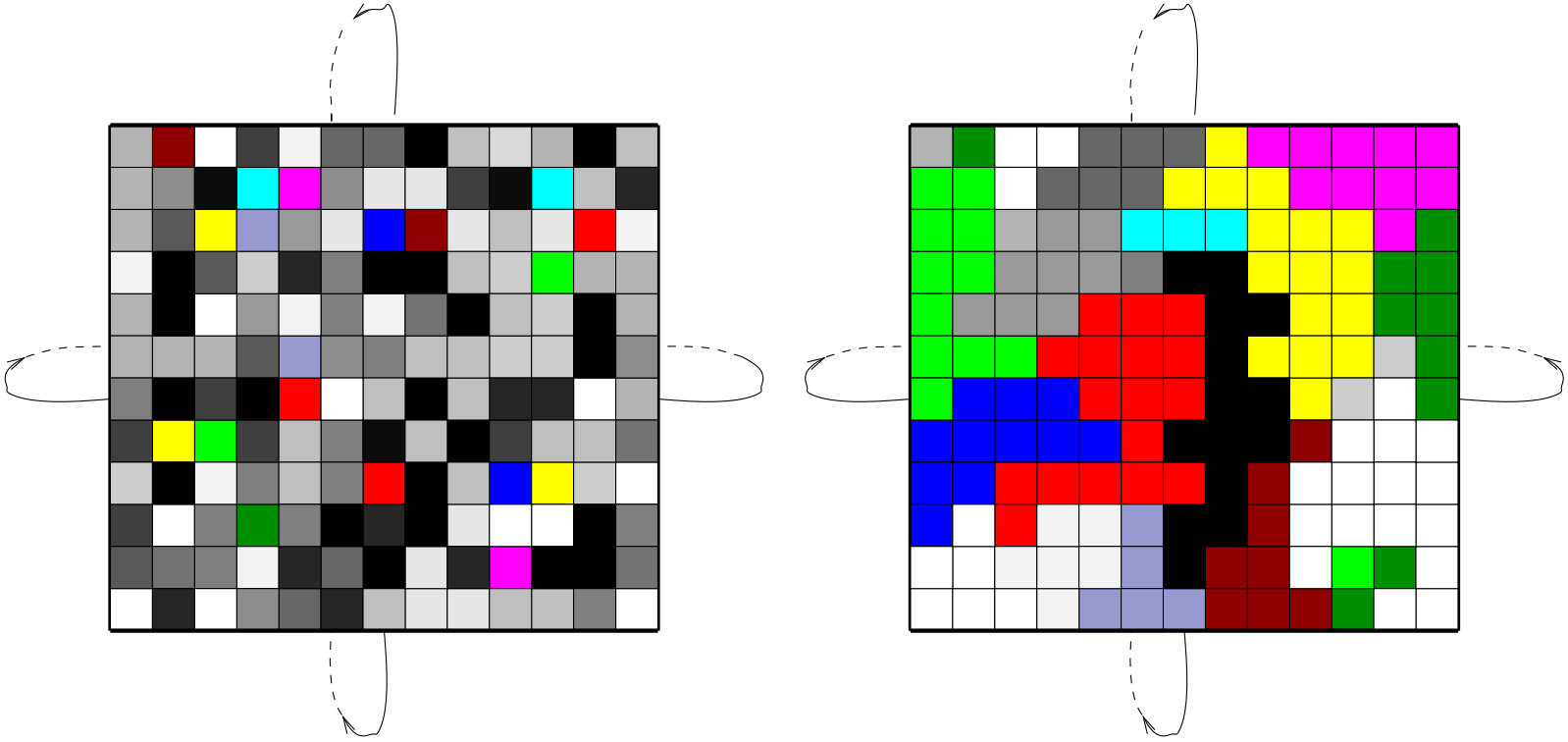
- Island Model Genetic Algorithm
  - Coarse Grained
- Cellular Genetic Algorithm
  - Fine Grained

ISLAND MODEL WITH MIGRATION





CELLULAR GENETIC ALGORITHM MODEL



Some No Free Lunch Results

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.

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

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

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

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

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
F2:  A C B
F3:  B A C
F4:  B C A
F5:  C A B
F6:  C B A
```

```
F1:  0 0 0 1
F2:  0 0 1 0
F3:  0 1 0 0
F4:  1 0 0 0
```

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).

QUESTION:

How should we evaluate search algorithms?

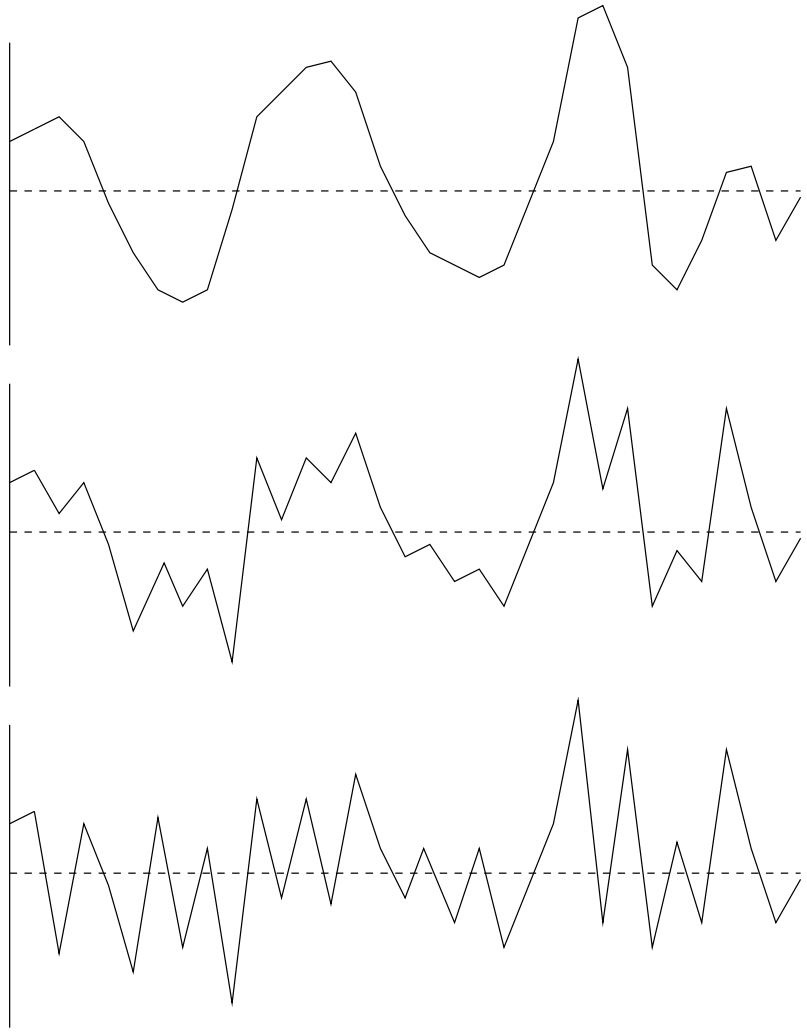
Let β represent a set of benchmarks.

$P(\beta)$ is the permutation closure over β.

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

S. Christensen and F. Oppacher

What can we learn from No Free Lunch? GECCO 2001



Gray vs Binary vs Real

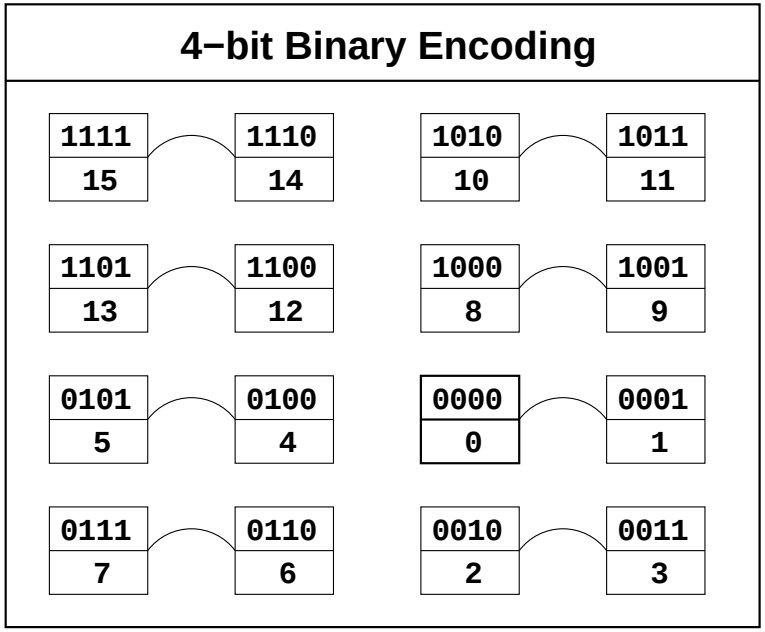
There are good arguments for Gray codes. The number of optima in Gray space are less than or equal to the number of optima in the "defining neighborhood" of Real Space.

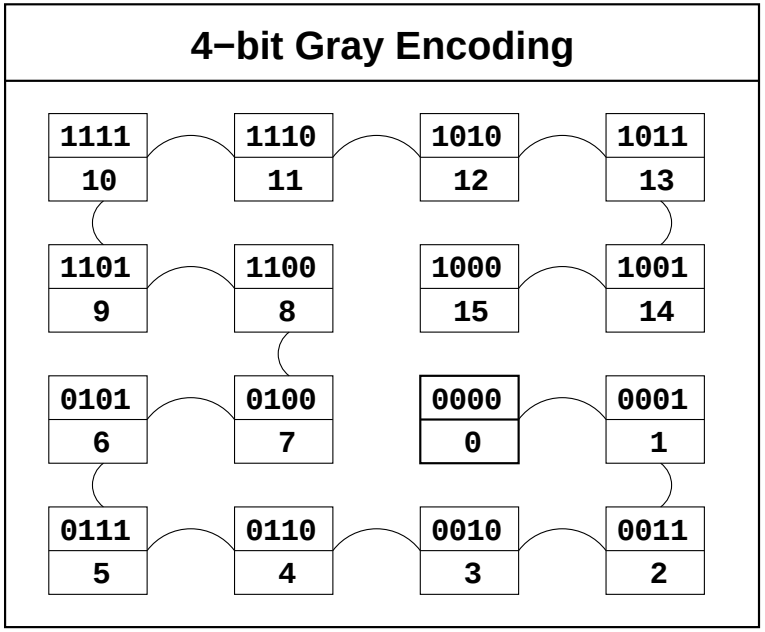
The number of local optima are important for local search methods.

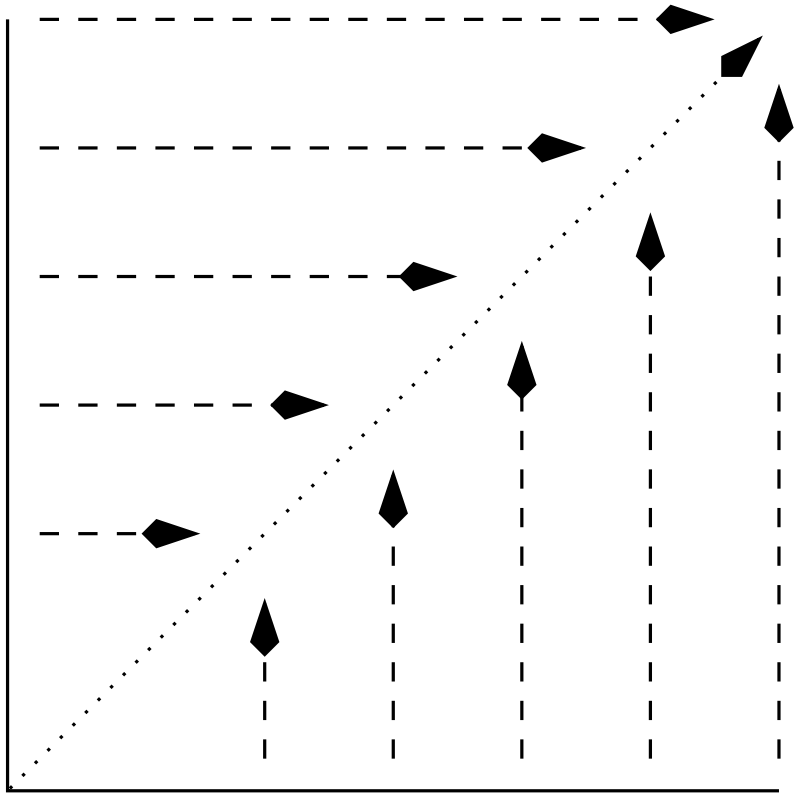
Comparisons that compare only Real and Binary
are less common (But still happen.)

Comparisons that involve different levels of bit precision are very common.

Adjacency


4−bit Binary Encoding
1111
15
1110
14
1010
10
1011
11
1101
13
1100
12
1000
8
1001
9
0101
5
0100
4
0000
0
0001
1
0111
7
0110
6
0010
2
0011
3

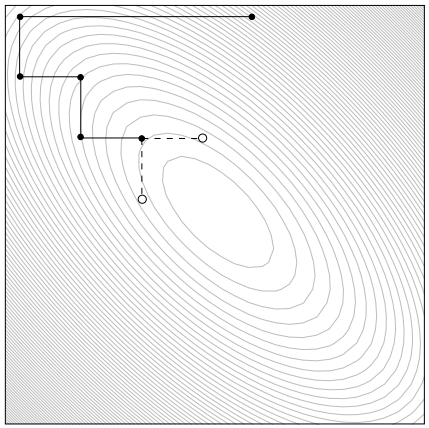

4−bit Gray Encoding
1111
10
1110
11
1010
12
1011
13
1101
9
1100
8
1000
15
1001
14
0101
6
0100
7
0000
0
0001
1
0111
5
0110
4
0010
3
0011
2

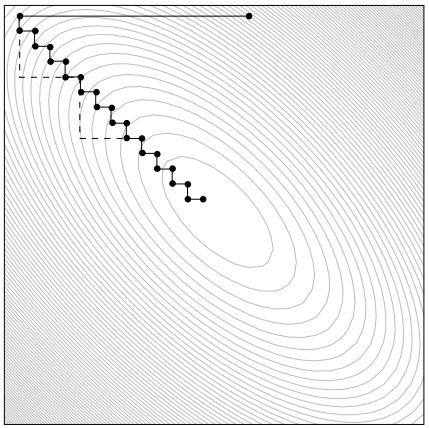


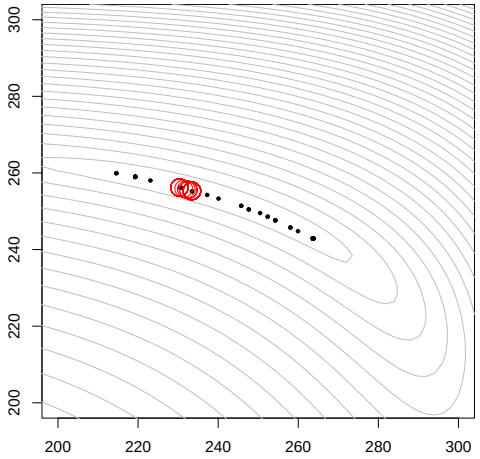
But Gray codes are “blind” to ridges.

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

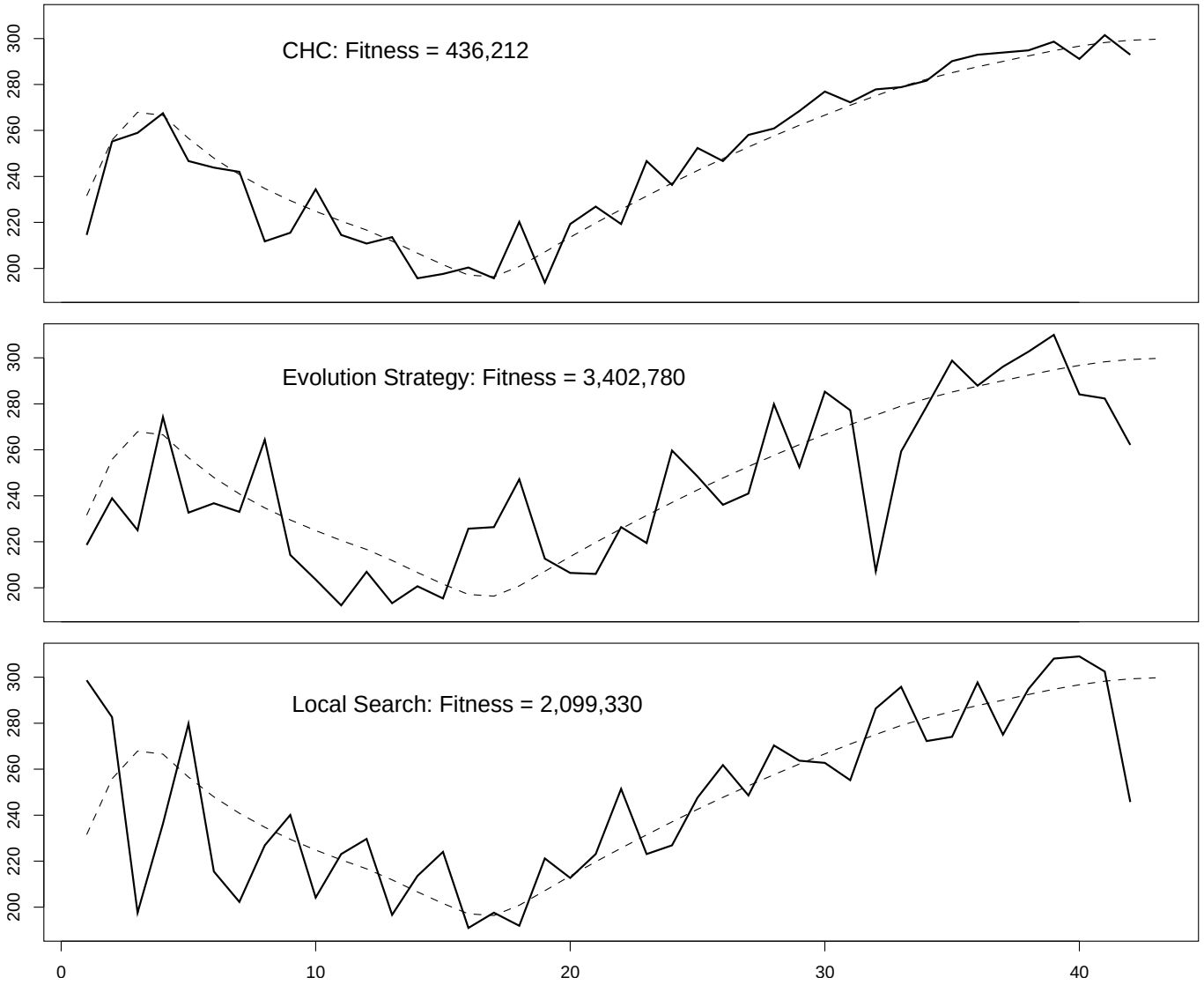
Researchers have created a *forward model* that relates 43 vertical temperature profiles ($\vec{x}$) to 2,000 observed measurements ($\vec{y}$).

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

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

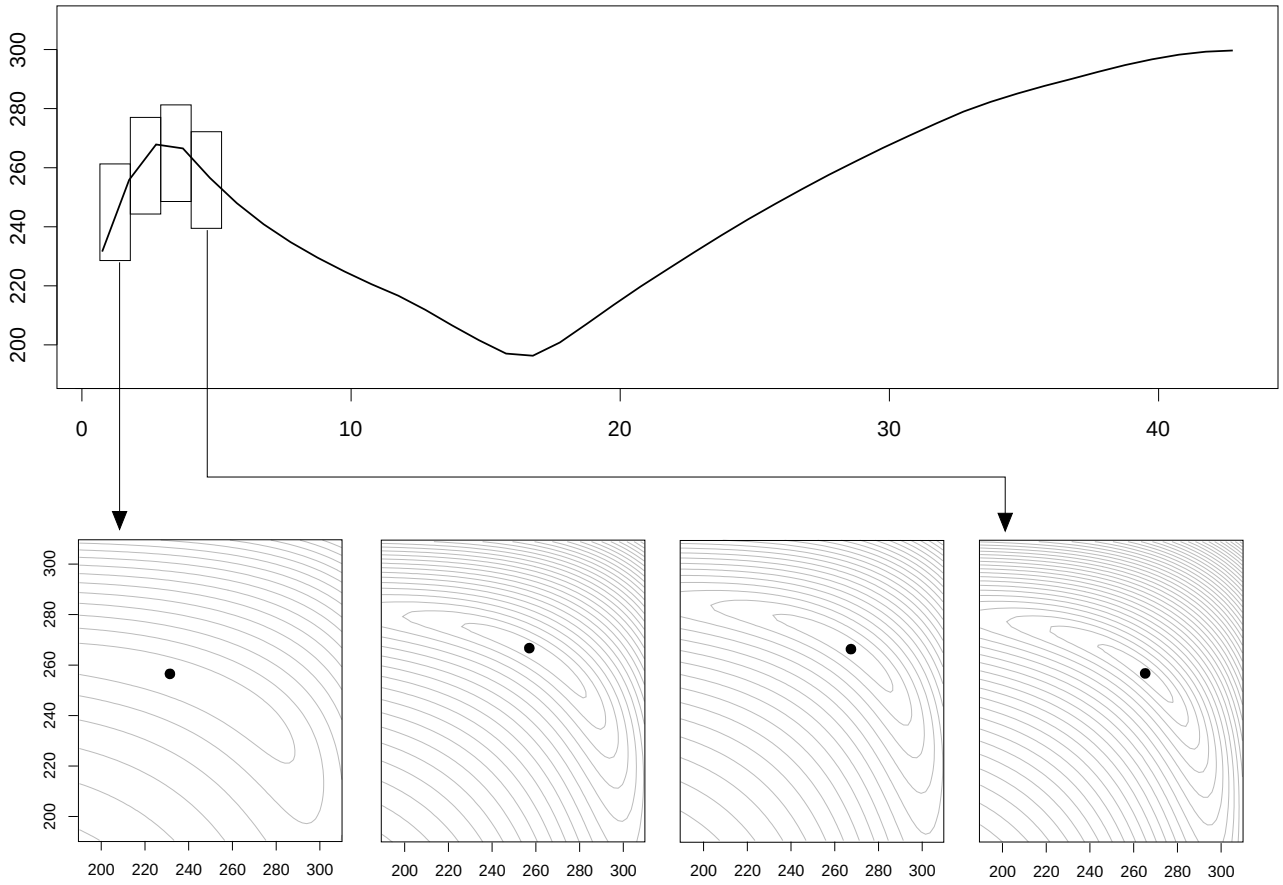
- Sometimes first order derivatives can be calculated analytically.
  In the general case, this is impossible.

Empirical Results



Why is the temperature problem so hard?

- Ridges in search space.



CMA Covariance Matrix Adaptation

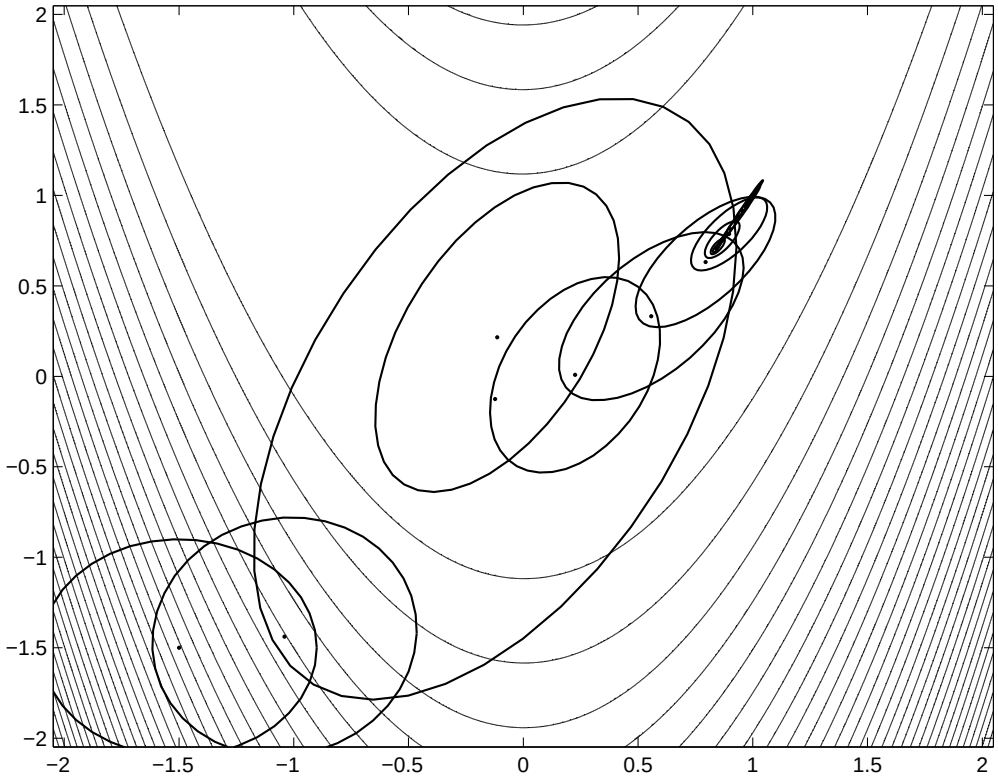
Let $\mathbf{Z}^{(g+1)}$ be the covariance of the μ best individuals.

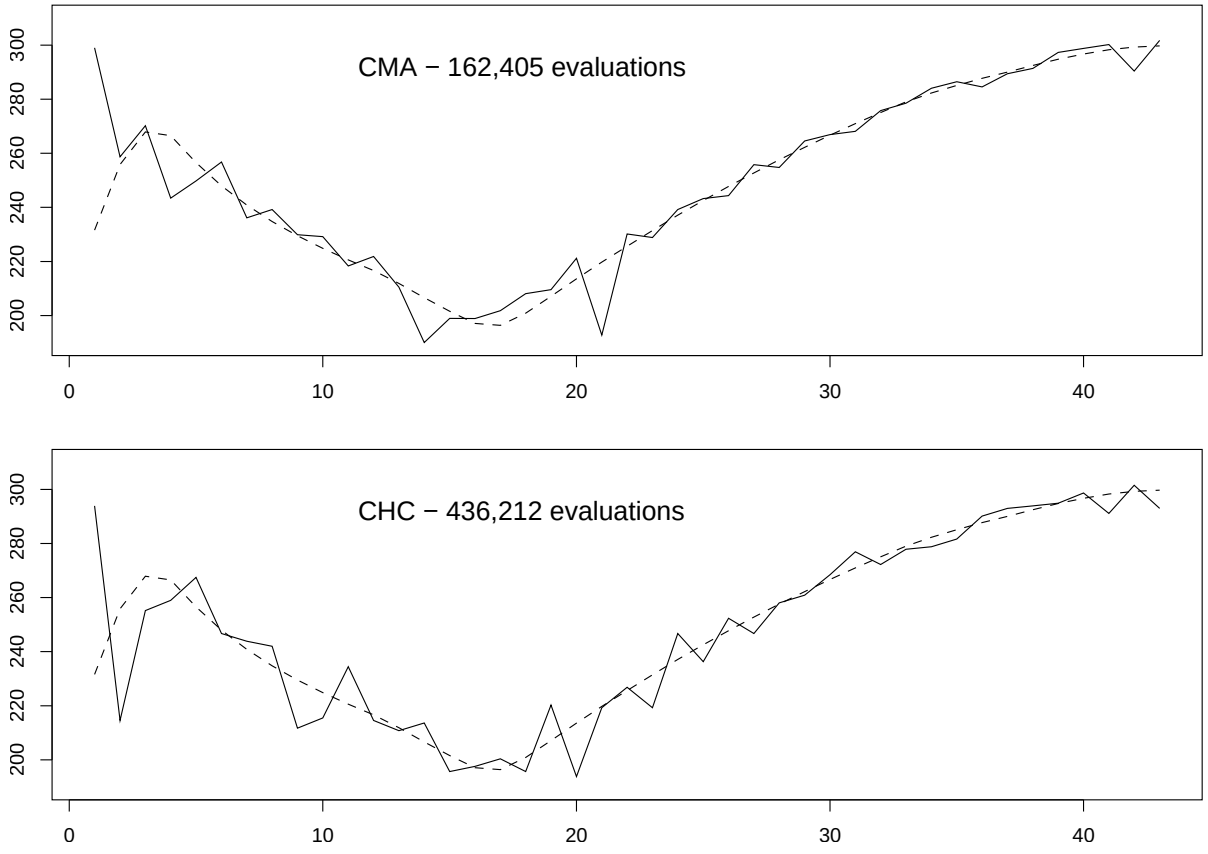
Let $\mathbf{P}^{(g+1)}$ be the covariance of the evolution path.

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.


2
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−2
−1.5
−1
−0.5
0
0.5
1
1.5
2


CMA − 162,405 evaluations
CHC − 436,212 evaluations
200
220
240
260
280
300
0
10
20
30
40

There is a fundamental tension is search between:

- Following the gradient to locate an optima
- Exploring as many optima as possible

"Exploration versus Exploitation"

"Intensification versus Diversification"

PRECISION plays a key role in this trade-off

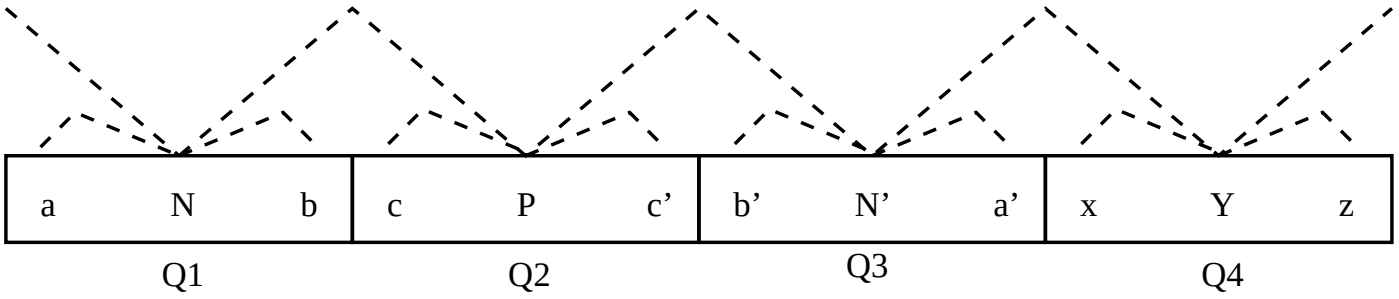
Comparing “Real-Valued” and “Bit” representation is much more complex than most of the literature suggests.

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

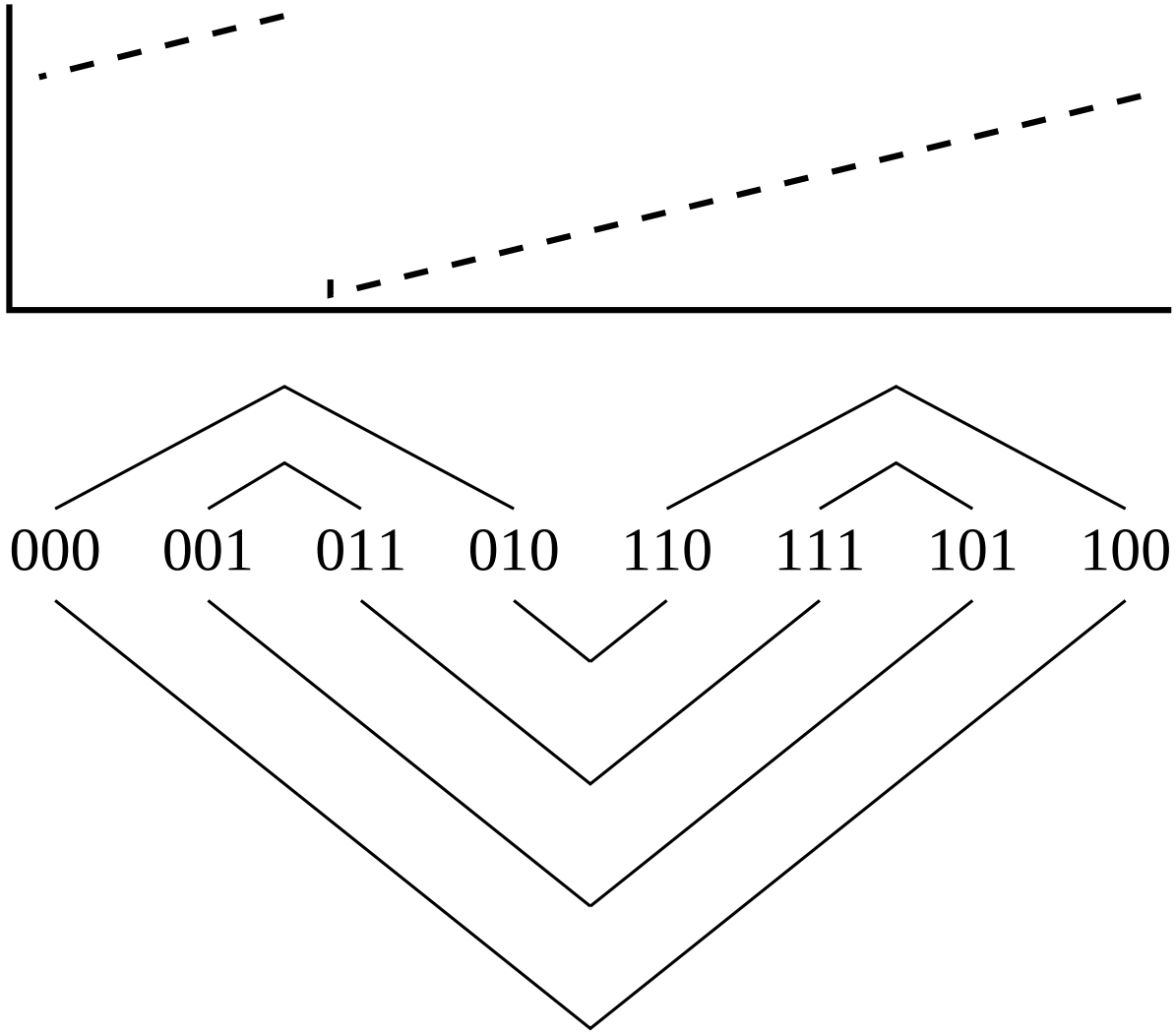
Low Precision might ”miss” good solutions.
But it aids exploration.

High Precision can result in low/slow exploration.

Local Quad Search



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


000
001
011
010
110
111
101
100

The Sphere Function

Algorithm	20-Dimension		30-D	
	Sol	Evals	Sol	Evals
Quad Search	30	1240	30	1890
Next Ascent	30	12115	30	18420
Steepest Asc	30	208198	30	458078
(50+50) ES	30	130571	28	500807

All searches at 32 bits of precision.

BUT did we use the right Evolution Strategy?? The 1/5 rule.

A Hybrid: Genetic Quad Search

We used GENITOR, a steady-state GA, as the genetic algorithm.

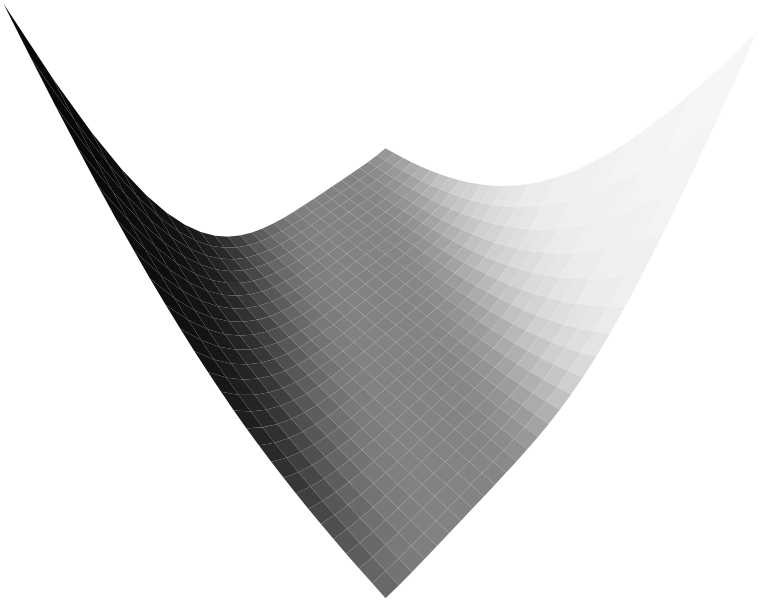
We also tested CHC and SGA .... not as good.

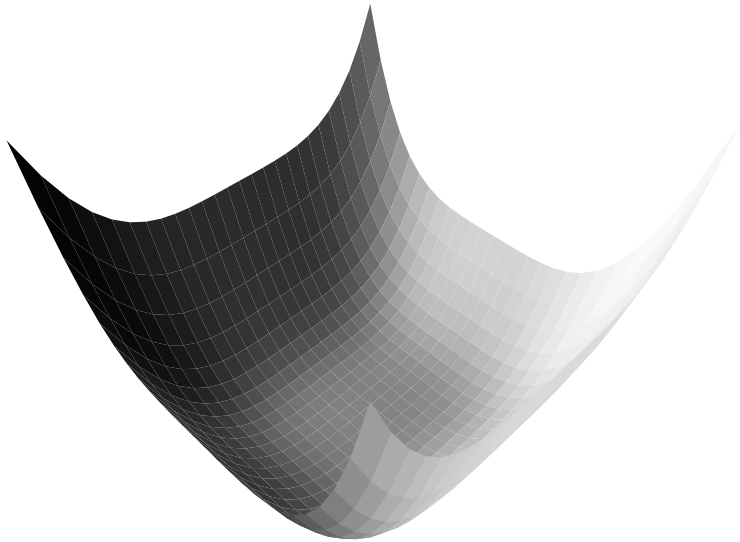
We used 3 forms of local search:

1. Quad Search
2. Local Search Bit Climbing (RBC, next ascent)
3. Steepest Ascent Bit Climbing (SABC)

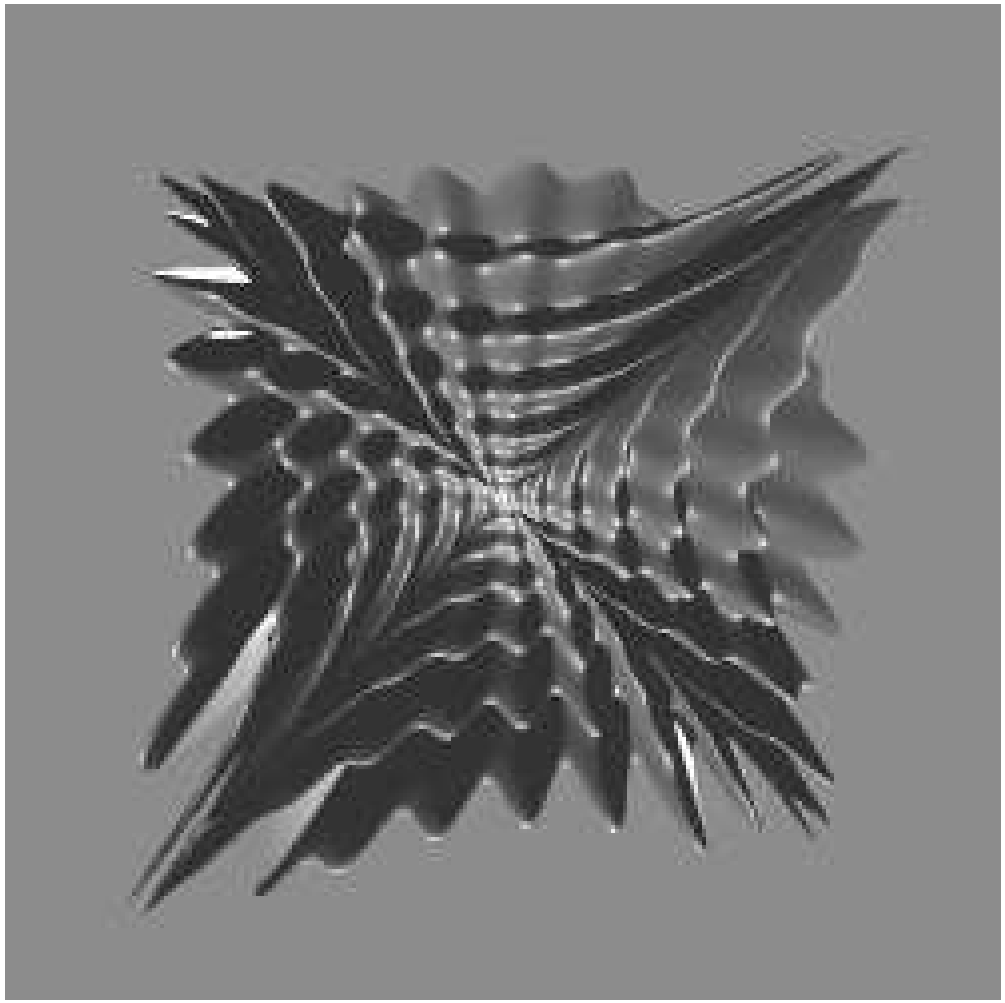
We ran RBC (and Steepest Ascent) in three modes:

1. Full (F): all strings improved with local search.
2. Stochastic (S): a string is improved with 5% probability.
3. Restricted (R): improve each string until 1 improvement is found.





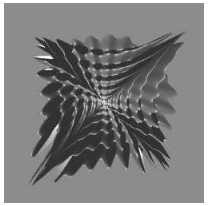
Powell $(x_1+10x_2)^2+(\sqrt{5}(x_3-x_4))^2+((x_2-2x_3)^2)^2+(\sqrt{10}(x_1-x_4)^2)^2$



Rana's Function $F(x,y) = xsin(\sqrt{|y+1-x|})cos(\sqrt{|x+y+1|})$
$+ (y+1)cos(\sqrt{|y+1-x|})sin(\sqrt{|x+y+1|})$

How do the hybrids do?

Function	ALG	Mean	σ	Sol	Evals
Powell	CHC	0	0.0	30	200K
Powell	Quad-F	3e-9	9e-9	22	262K
Powell	RBC-F	2e-4	8e-5	0	—
Powell	RBC-S	1e-7	7e-7	5	351K
Rana	CHC	-495.5	5.5	0	—
Rana	Quad-F	-510.2	2.5	26	267K
Rana	RBC-F	-471.4	7.0	0	—
Rana	RBC-S	-484.0	7.6	0	—



Function	Algorithm	Mean	σ	Solved	Evals
Rana 10-D	CMA-ES	-388.0	15.0	0	500K
	Quad	-434.8	8.4	0	500K
	RBC	-446.4	9.9	0	500K
	Genitor	-443.4	17.8	0	500K
	CHC	-495.5	5.5	0	500K
	Hybrid Quad	-510.3	2.5	26	268K

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.

Local Search for Permutations

Consider the following jobs to be scheduled

```
D J K G C N A B E M F H L I
```

Let the move operator be a “shift operator”.

Pick a *job* to move.

Pick a *location* after some other job.

EXAMPLE: Move K After B

```
D J K G C N A B E M F H L I
    - <--------

D J G C N A B K E M F H L I
    <-------- -
```

EXAMPLE: Move B After K

```
D J K G C N A B E M F H L I
      ------> -

D J K B G C N A E M F H L I
      - ------>
```

Complexity of 1-move is $O(N^2)$

TABU SEARCH

Brute Force: the last $|T|$ solutions are tabu.
prevents cycles of $|T|$ or less.

```
Assume we ...  Move B ... After K


TABU:  B cannot be moved again for 5 steps


TABU:  Nothing can move after K for 5 steps


TABU-LIST:          Tabu Moves:  B > F > M > G > L
                 Tabu Locations:  K > D > I > J > C
```

REACTIVE TABU SEARCH

Adapts search parameters based on recent history.

1. *ADAPTIVE PROHIBITION:*
   The length of the tabu list (i.e., the prohibition time, T) is determined through feedback mechanisms during the search.

   T is increased when diversification is needed;
   (repetition of previously-visited points)
   it decreases when this need disappears.

2. *ESCAPE:*
   A number of random moves away from the current point.

3. *FAST MEMORY (HISTORY):*
   To store previous-visited points.

An online tutorial and key papers can be found at

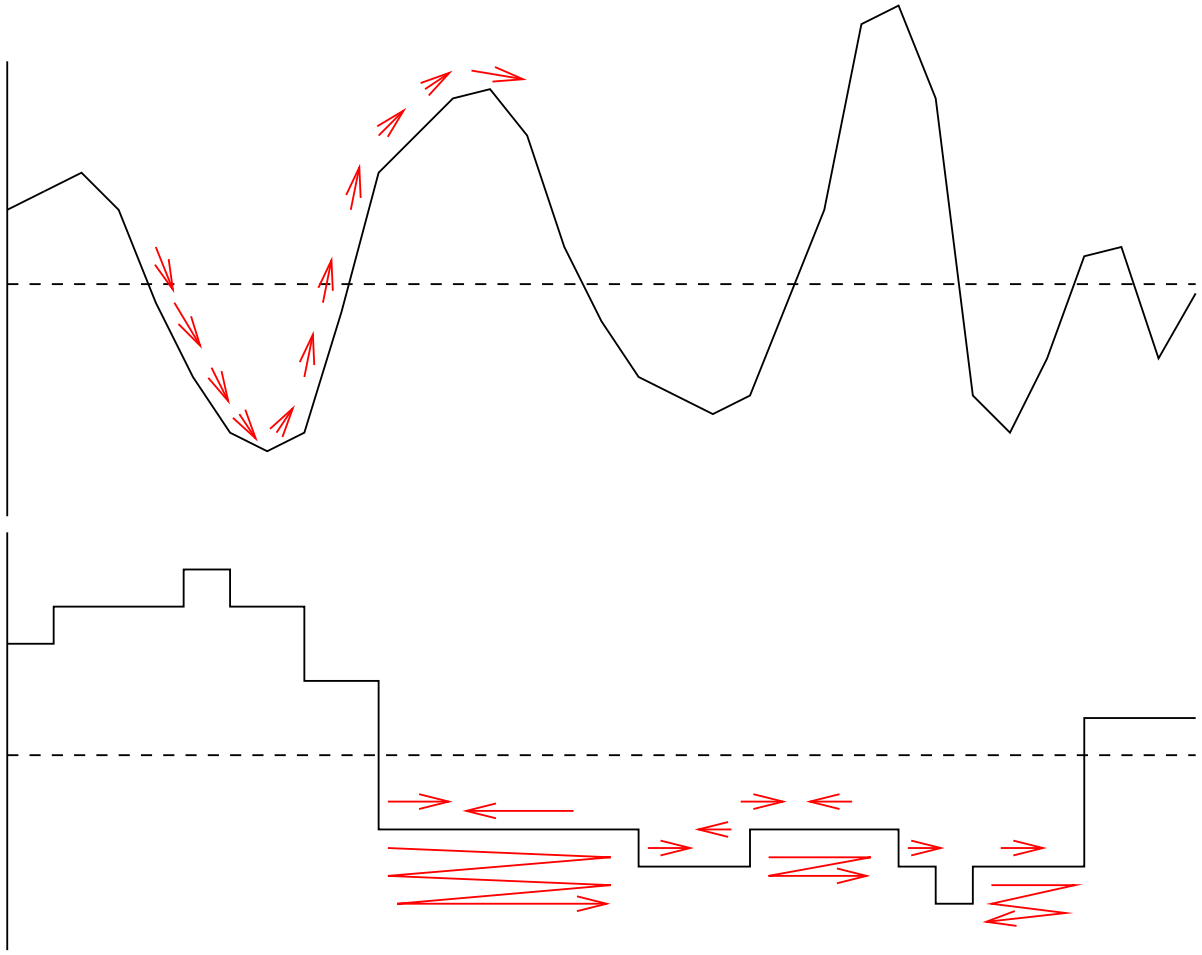
http:rtm.science.unitn.it/∼battiti/tutorial/tutorial.html

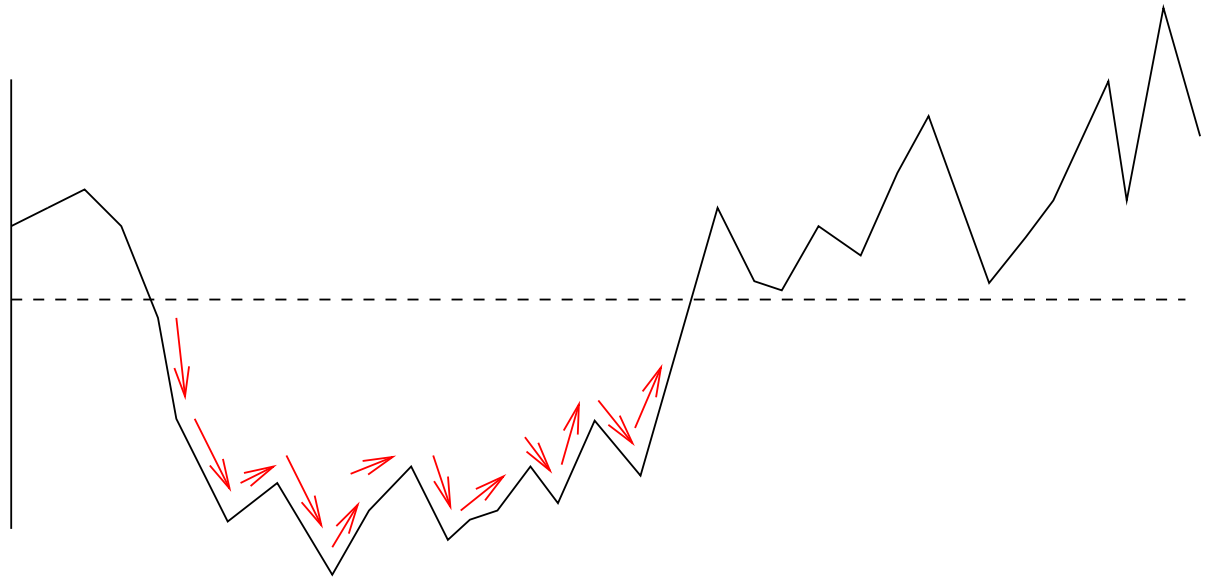
TABU SEARCH often does not work well
for parameter optimzation problems.

Is also doesn't seem to help (much)
with classic problems like MAXSAT.

Works extremely well for some scheduling application.

Why?





KEY ISSUES:

1. Good local minima are very near other good local minima
2. The neighborhood size must be pruned.

Complexity of $O(N^2)$ is too much.
A critical path neighborhood is used for Job Shop Scheduling.

SO:

Tabu Search works well when the neigborhood is restricted

AND

Short uphill moves finds new basins of attraction.

On some scheduling problems where these things are not true,
Genetic Algorithms out-performs Tabu Search.

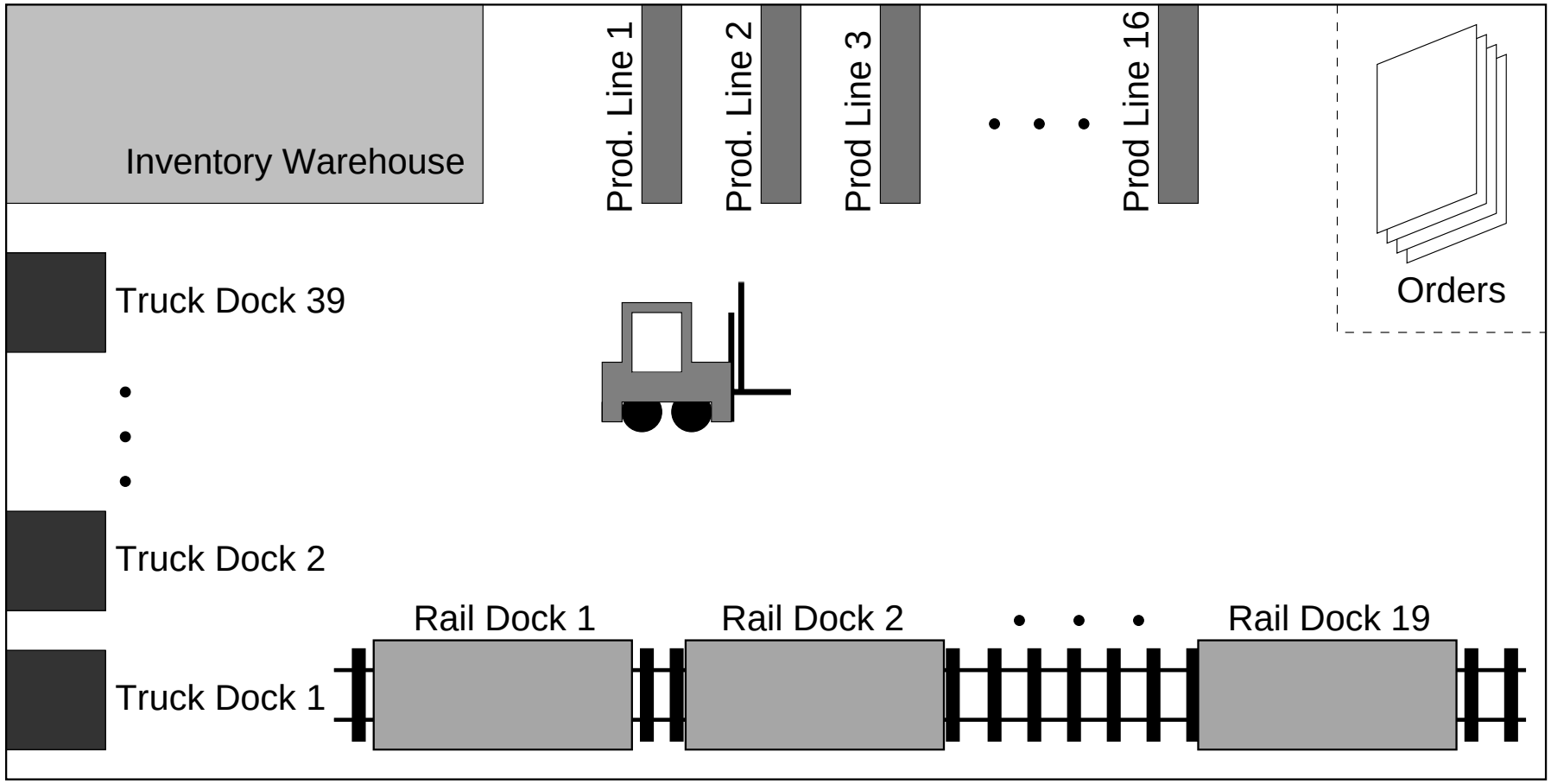
Examples:

Warehouse Scheduling and Satellite Scheduling

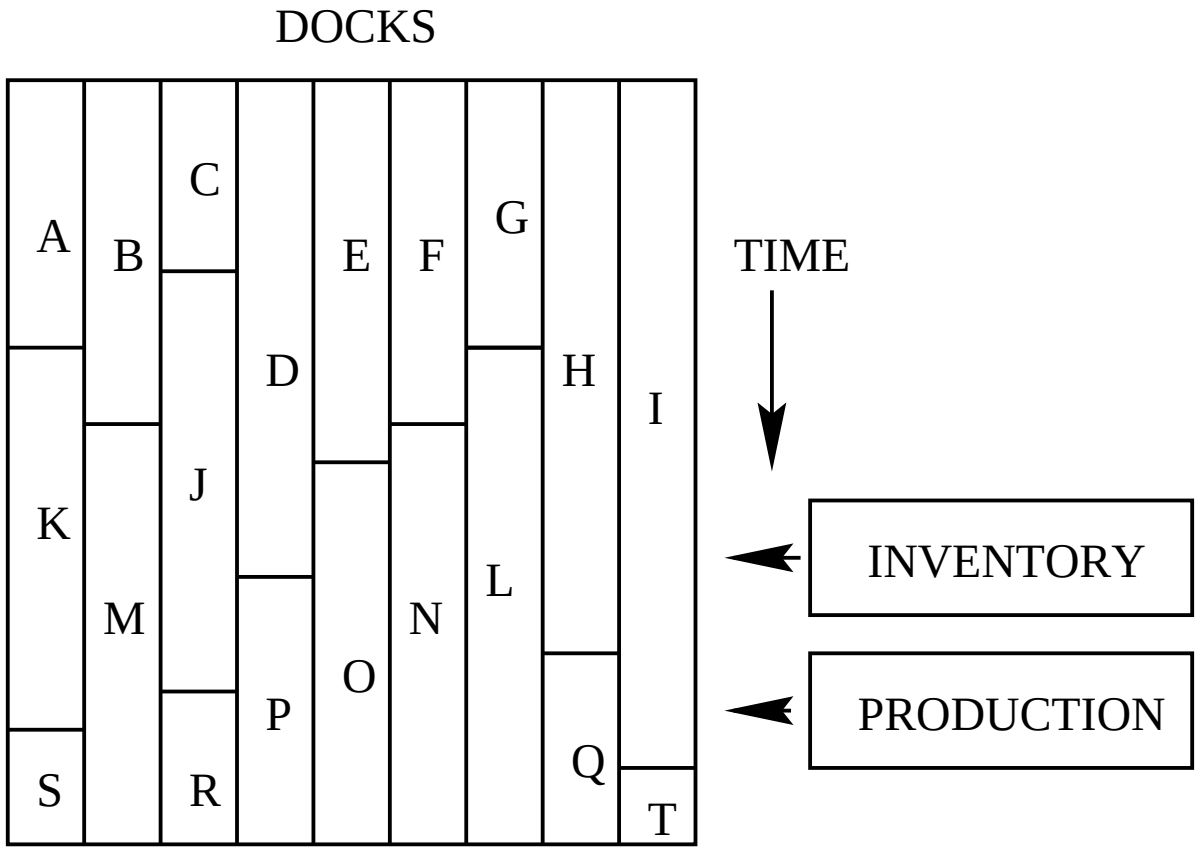
Syswerda's Order Crossover

```
Parent 1:   A B C D E F G H I J K
              * *   * *   *
Parent 2:   C F H A K B E J D I G

             Parent 1 chosen

Offspring:  A C F D H B G E I J K
```

Order Crossover was used in the GENITOR algorithm.


Inventory Warehouse
Prod. Line 1
Prod. Line 2
Prod Line 3
Prod Line 16
Orders
Truck Dock 39
Truck Dock 2
Truck Dock 1
Rail Dock 1
Rail Dock 2
Rail Dock 19

Customer Priority Queue: A, B, C, D, E, F, G, H, I, ..., Z

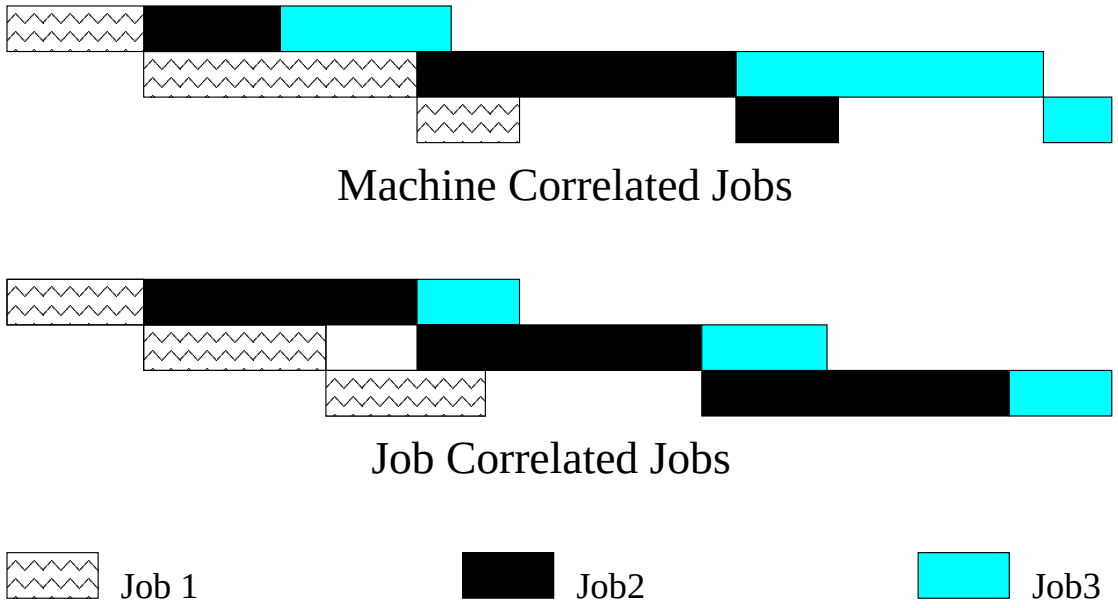


The Objective Function

- *mean-time-at-dock*
- *average-inventory*
- Combination of *mean-time-at-dock* and *average-inventory* [a]

$$obj = \frac{(ai - \mu_{ai})}{\sigma_{ai}} + \frac{(mt - \mu_{mt})}{\sigma_{mt}}$$

[a] Bresina, Drummond and Swanson, "Expected Solution Quality", IJCAI 1995

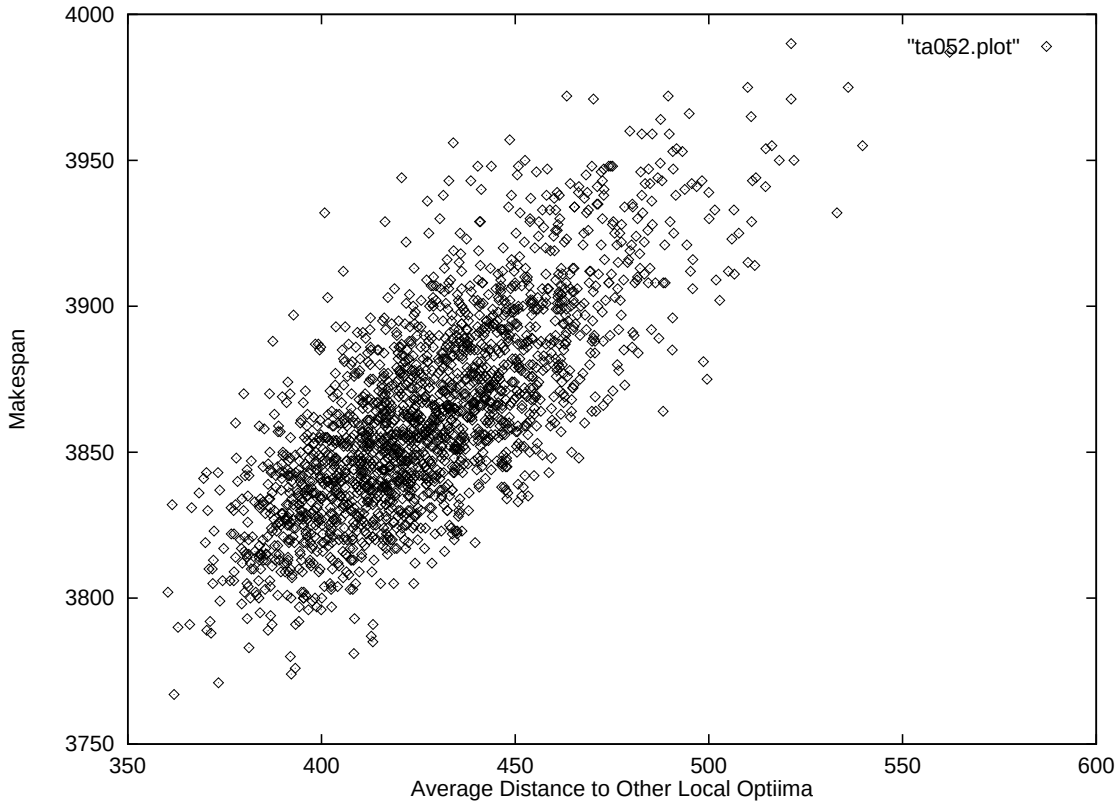


The PERMUTATION FLOWSHOP SCHEDULING PROBLEM.

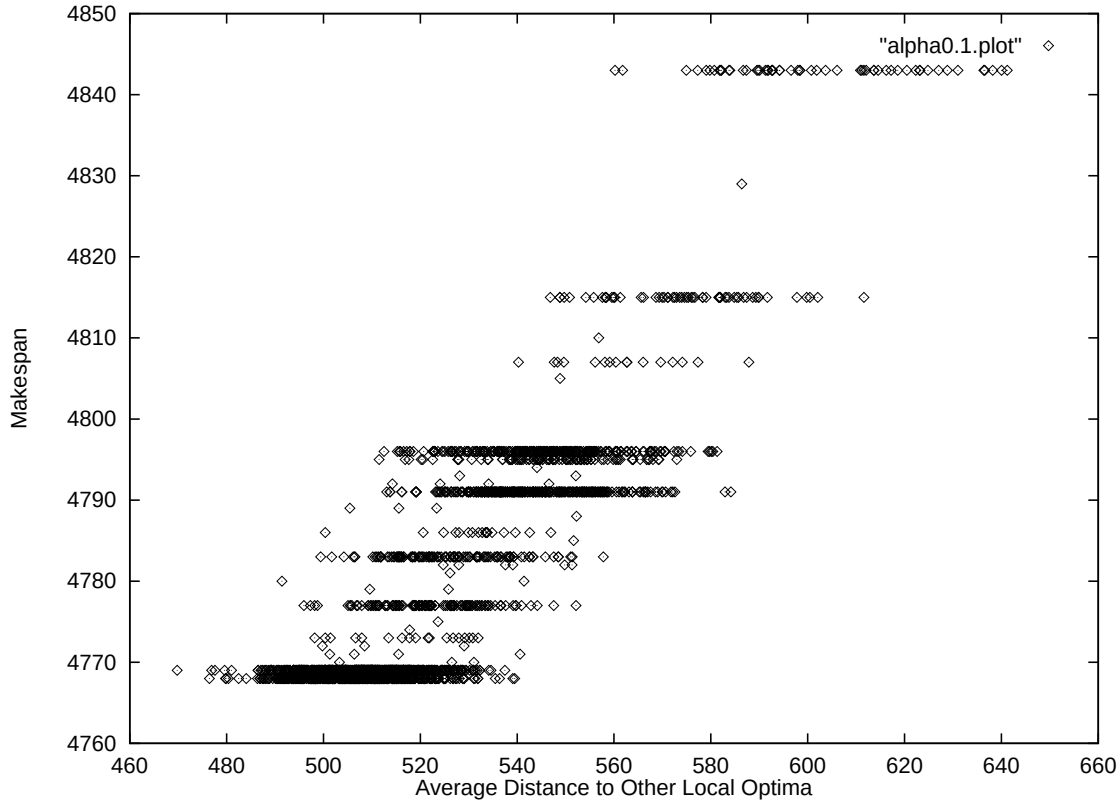
Benchmark are typically generated randomly.

Real-world problems may have correlated structure.

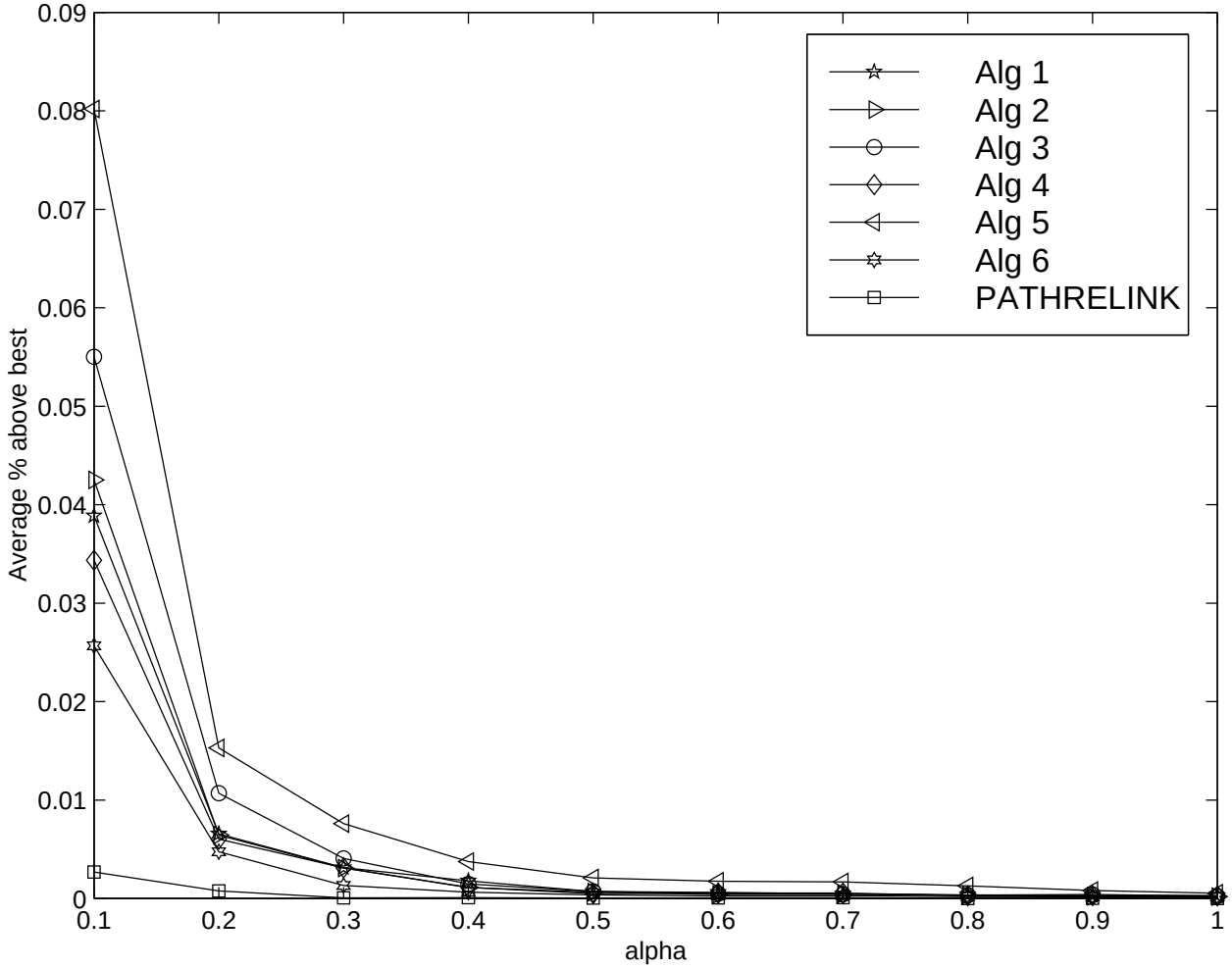
Job could be *machine correlated* or *job correlated*.



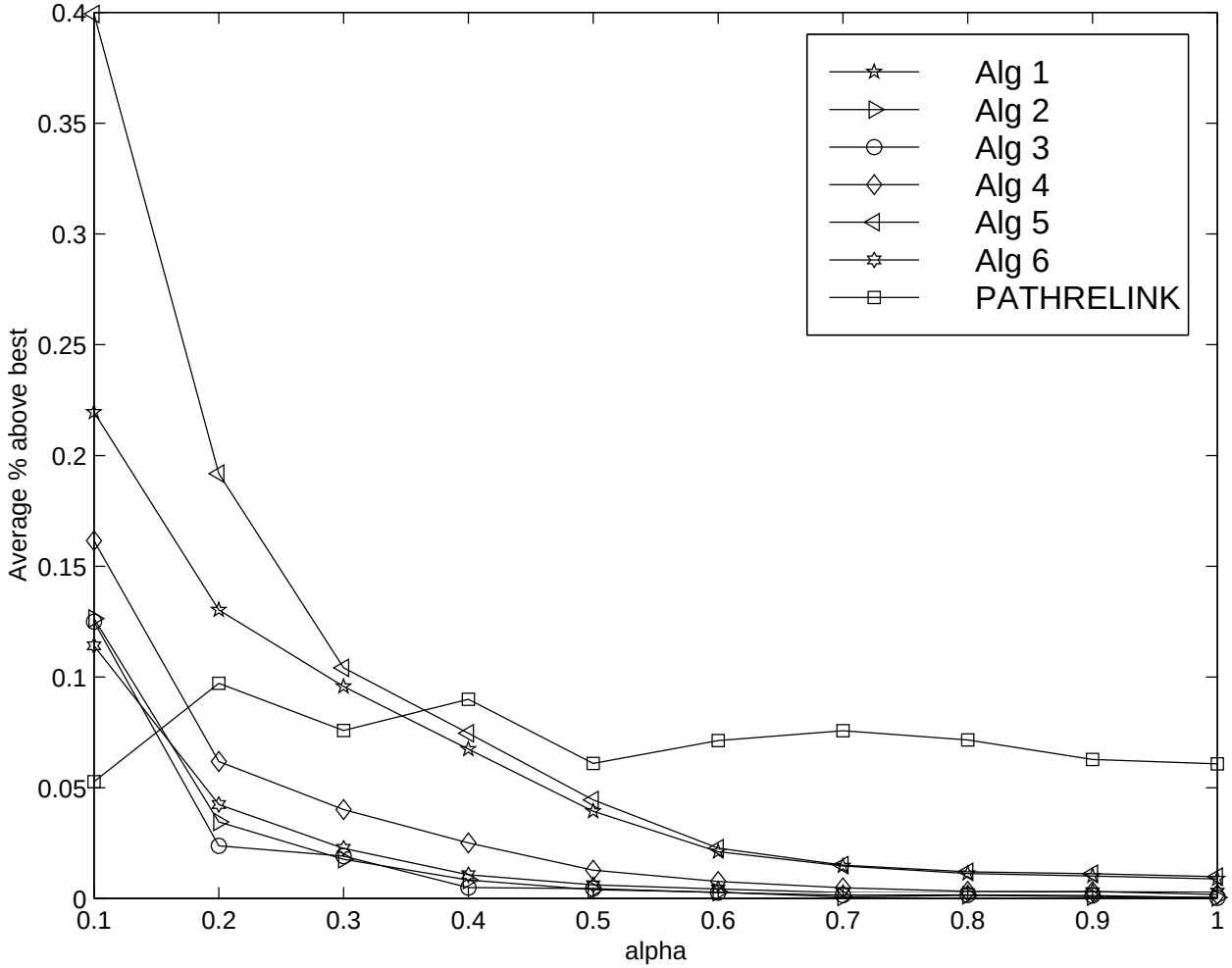
The BIG VALLEY effect



For Correlated Problems, the “Big Valley” looks different.



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.