A Theoretical & Empirical Analysis of Evolutionary Testing and Hill Climbing for Structural Test Data Generation

ISSTA July 2007

Mark Harman
King's College London

Phil McMinn
Sheffield University

Which Search Technique: Global or Local?

Mark Harman
King's College London

Phil McMinn
Sheffield University

Which Search Technique: Global or Local?

Mark Harman and Phil McMinn.
A Theoretical and Empirical Analysis of Evolutionary Testing and Hill Climbing for Structural Test Data Generation
ISSTA 2007.

Mark Harman and Phil McMinn,
A Theoretical and Empirical Study of Search Based Testing:
Local, Global and Hybrid Search
TSE. To appear.

Which Search Technique: Global or Local?

Mark Harman and Phil McMinn.
A Theoretical and Empirical Analysis of Evolutionary Testing and Hill Climbing for Structural Test Data Generation
ISSTA 2007.

Mark Harman and Phil McMinn,
A Theoretical and Empirical Study of Search Based Testing:
Local, Global and Hybrid Search
TSE. To appear.

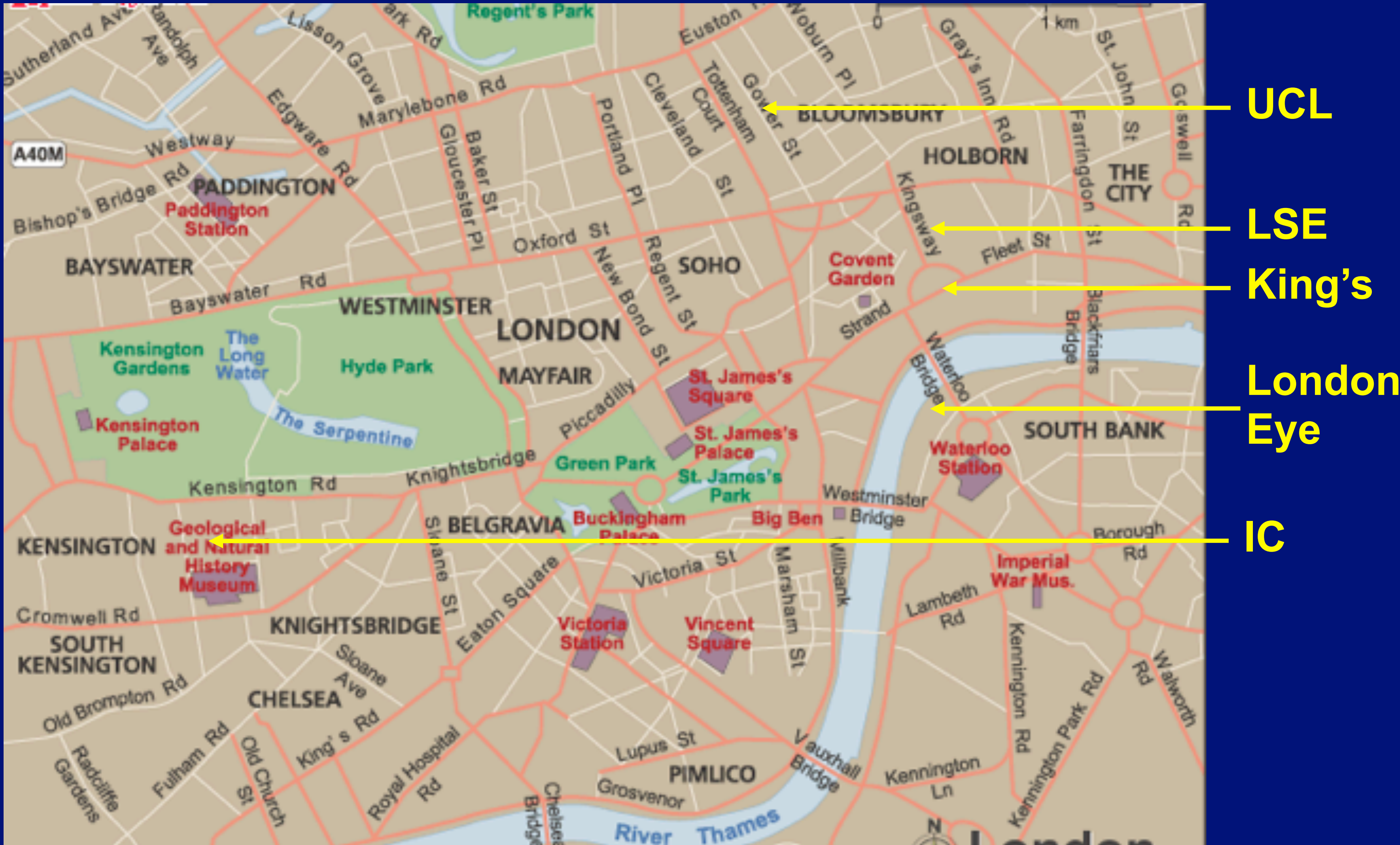
Which Search Technique: Global or Local?

Mark Harman and Phil McMinn.
A Theoretical and Empirical Analysis of Evolutionary Testing and Hill Climbing for Structural Test Data Generation
ISSTA 2007.

Mark Harman and Phil McMinn,
A Theoretical and Empirical Study of Search Based Testing:
Local, Global and Hybrid Search
TSE. To appear.

Author order is alphabetical

Where is King's College London?

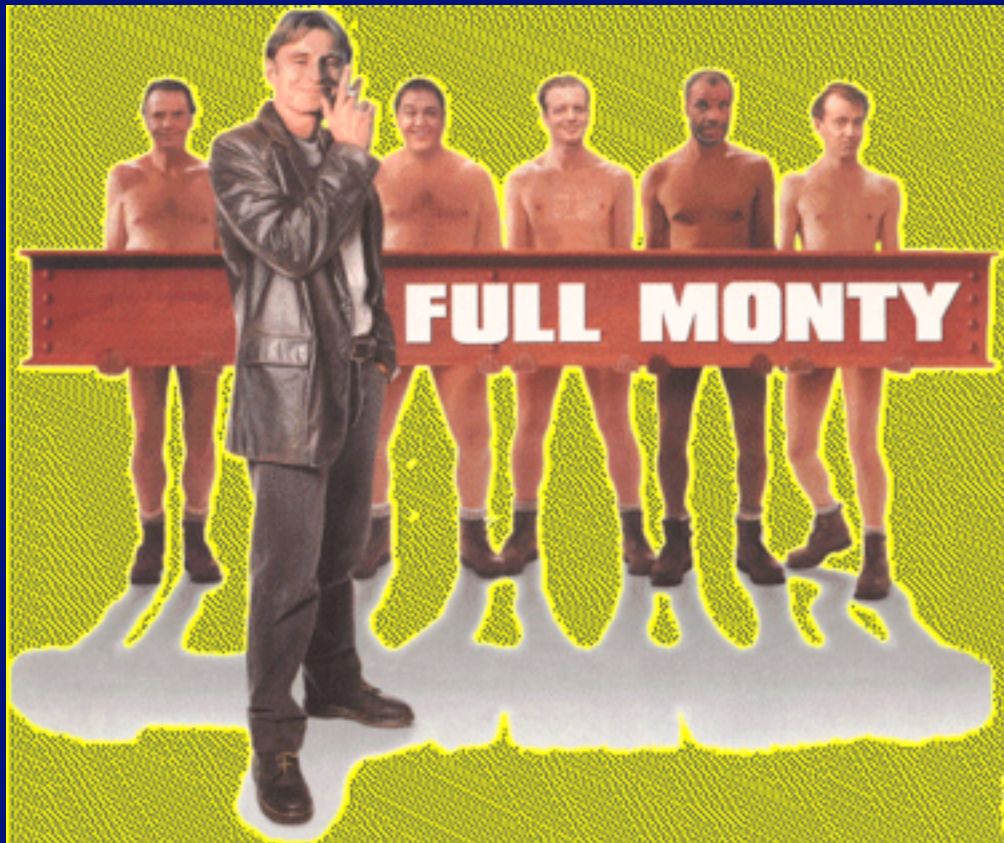


Where is Sheffield University?

Sheffield

London

No Full Monty Joke



No Full Monty Joke

Sorry

Overview

Search Based Testing

- **Local**: Hill Climbing using Alternating variable method
- **Global**: Genetic Algorithms

Theoretical foundations

- Schemas
- Royal Roads

Empirical study

Implications

What is SBT

In Search based testing we apply search techniques to search large input spaces, guided by a fitness function.

What is SBT

In Search based testing we apply search techniques to search large input spaces, guided by a fitness function.

Genetic Algorithms, Hill climbing, Simulated Annealing, Random, Tabu Search, Estimation of Distribution Algorithms, Particle Swarm Optimization

What is SBT

In Search based testing we apply search techniques to search large input spaces, guided by a fitness function.

Genetic Algorithms, Hill climbing, Simulated Annealing, Random, Tabu Search, Estimation of Distribution Algorithms, Particle Swarm Optimization

What is SBT

- In Search based testing we apply search techniques to search large input spaces, guided by a fitness function.
- Genetic Algorithms, Hill climbing, Simulated Annealing, Random, Tabu Search, Estimation of Distribution Algorithms, Particle Swarm Optimization

Structural Testing

Focus on branch testing

- Most widely studied
- So ready for some more in depth analysis

Other Search Based Testing Applications

Temporal Wegener et al.

Coverage Pargass & Harrold, Xanthakis et al., McMinn, Harman, Michael et al, Sthamer, Jones …

Functional Wegener et al.

Regression Rothermel et al., Woolcott et al., Yoo and Harman,…

Interaction Cohen et al. Bryce, Colbourn

Exception Tracey and Clark

Stress Briand et al., Antoniol, Di Penta

Robustness Shultz et al.

Structural Testing

- Focus on branch testing
  - Most widely studied
  - So ready for some more in depth analysis
- Two algorithms:
  - Hill Climbing, using Korel's alternating variable method
  - Genetic Algorithms, using DaimlerChrysler approach

Structural Testing

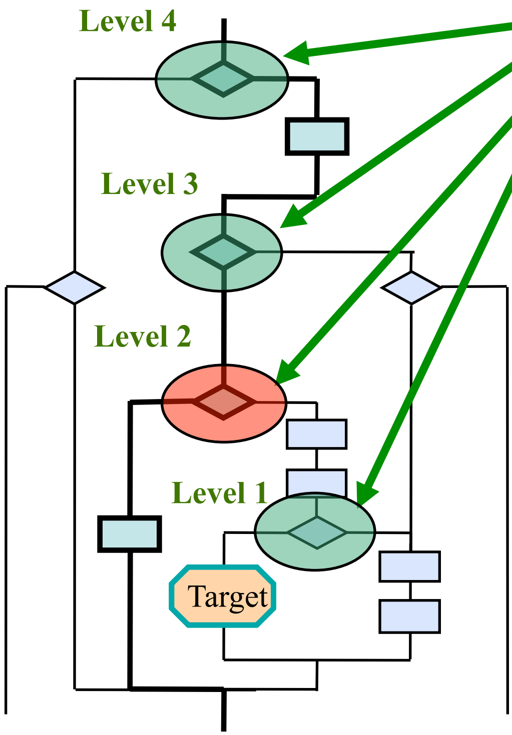
Focus on branch testing

- Most widely studied
- So ready for some more in depth analysis

Two algorithms:

- Hill Climbing, using Korel's alternating variable method
- Genetic Algorithms, using DaimlerChrysler approach
- … and Random Search

Fitness Computation



1. Approximation level

Identify relevant branching statements using control dependence

Evaluation of predicate in a branching condition

if A = B   Local_Distance = | A – B |

Fitness = Approximation_Level + Local_Distance

Alternating Variable Method

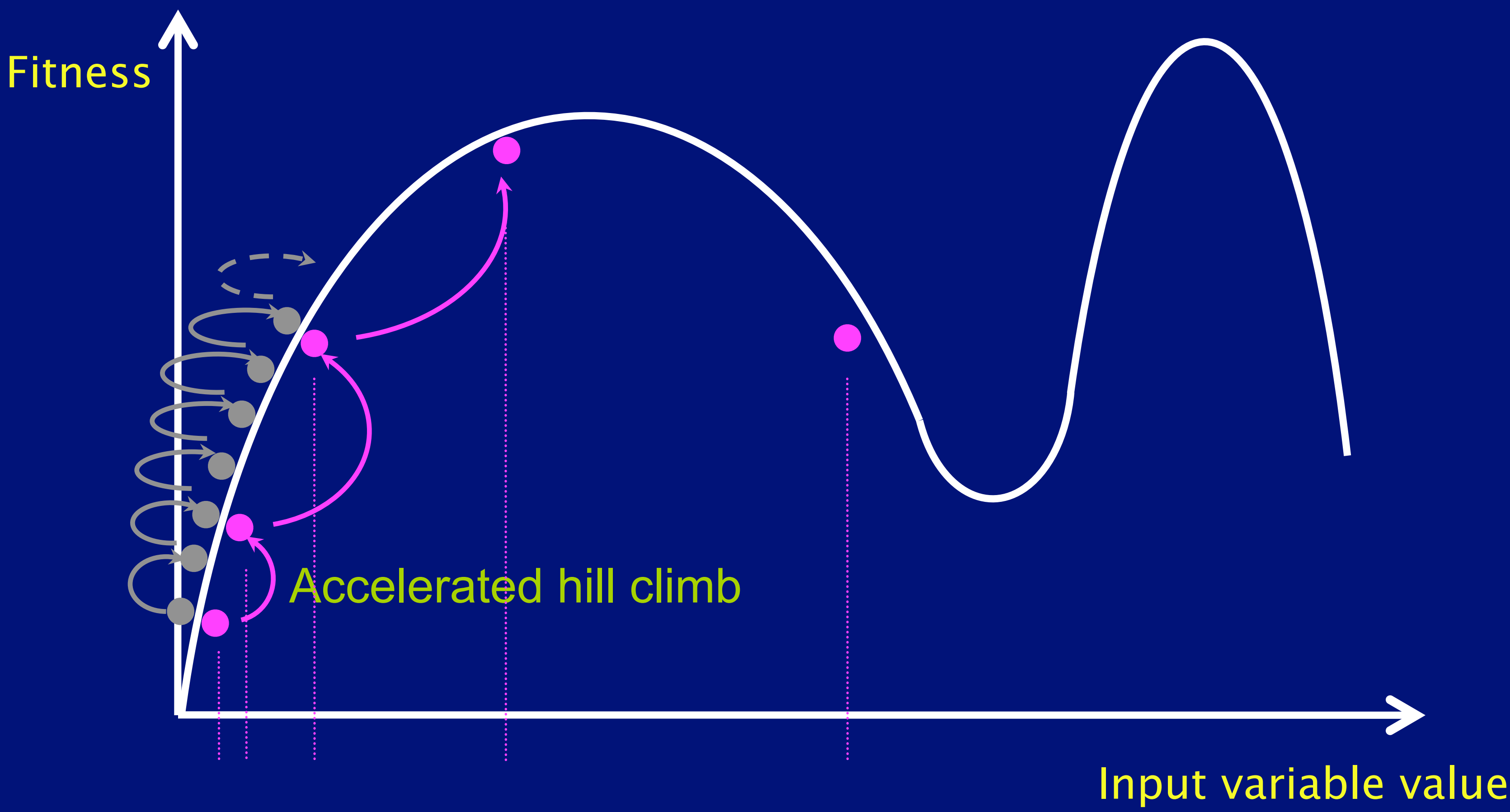
The alternating variable method is
hill climbing plus accelerated moves

Near Neighbour?
One small increase
One small decrease
} For some input variable

Method:
Cycle through input variables one at a time:
probe moves move to near neighbour:
If probing works, make accelerated pattern moves
Until no improvement on any variable

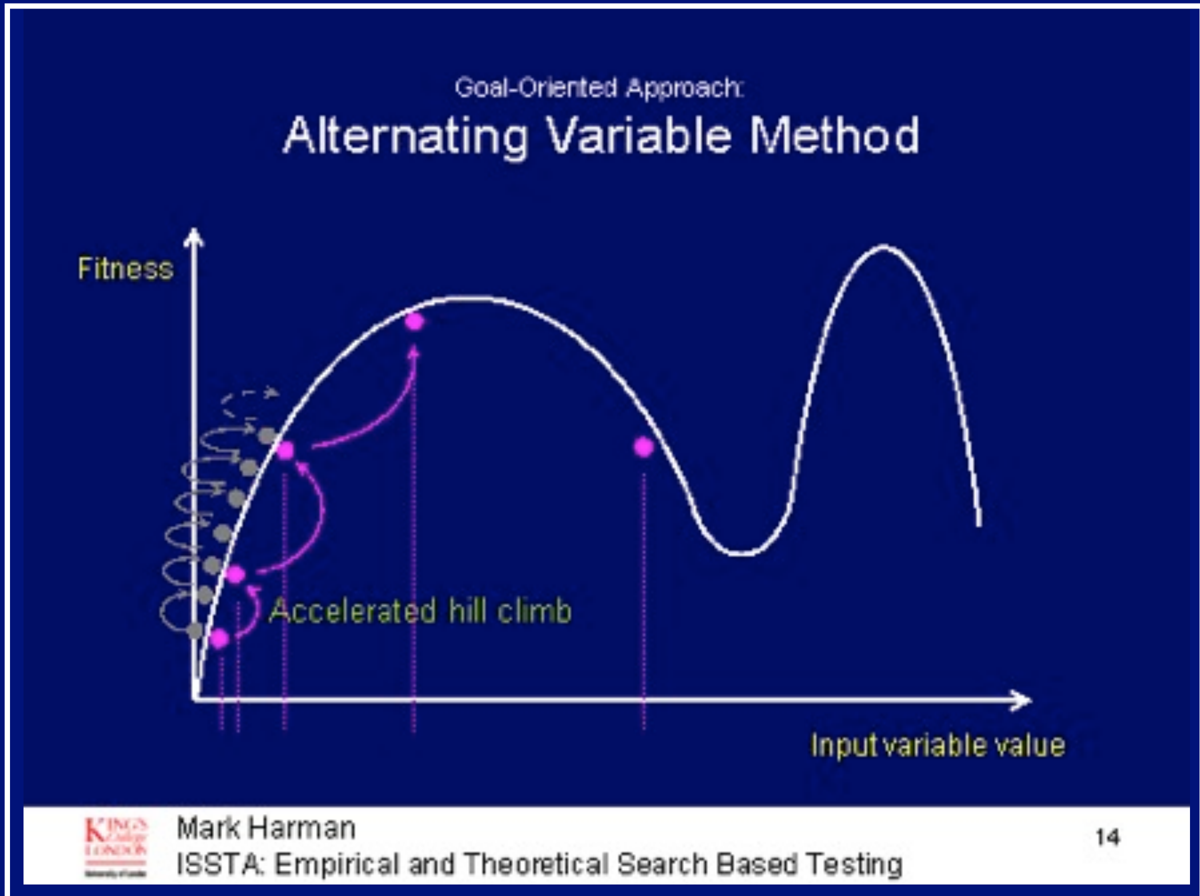
Goal-Oriented Approach:

Alternating Variable Method



Hill Climbing ↔ Steepest Descent

Hill Climbing ↔ Steepest Descent



Alternating Variable Method Example

```
void example(int a, int b, int c) {
    if (a == 0) {
        ...
    }

    if (b == 0) {
        if (c == 0) {
            // target
        }
    }
}
```

Random start: a=10 b=20 c=30

Case a :-

Probe move has no effect

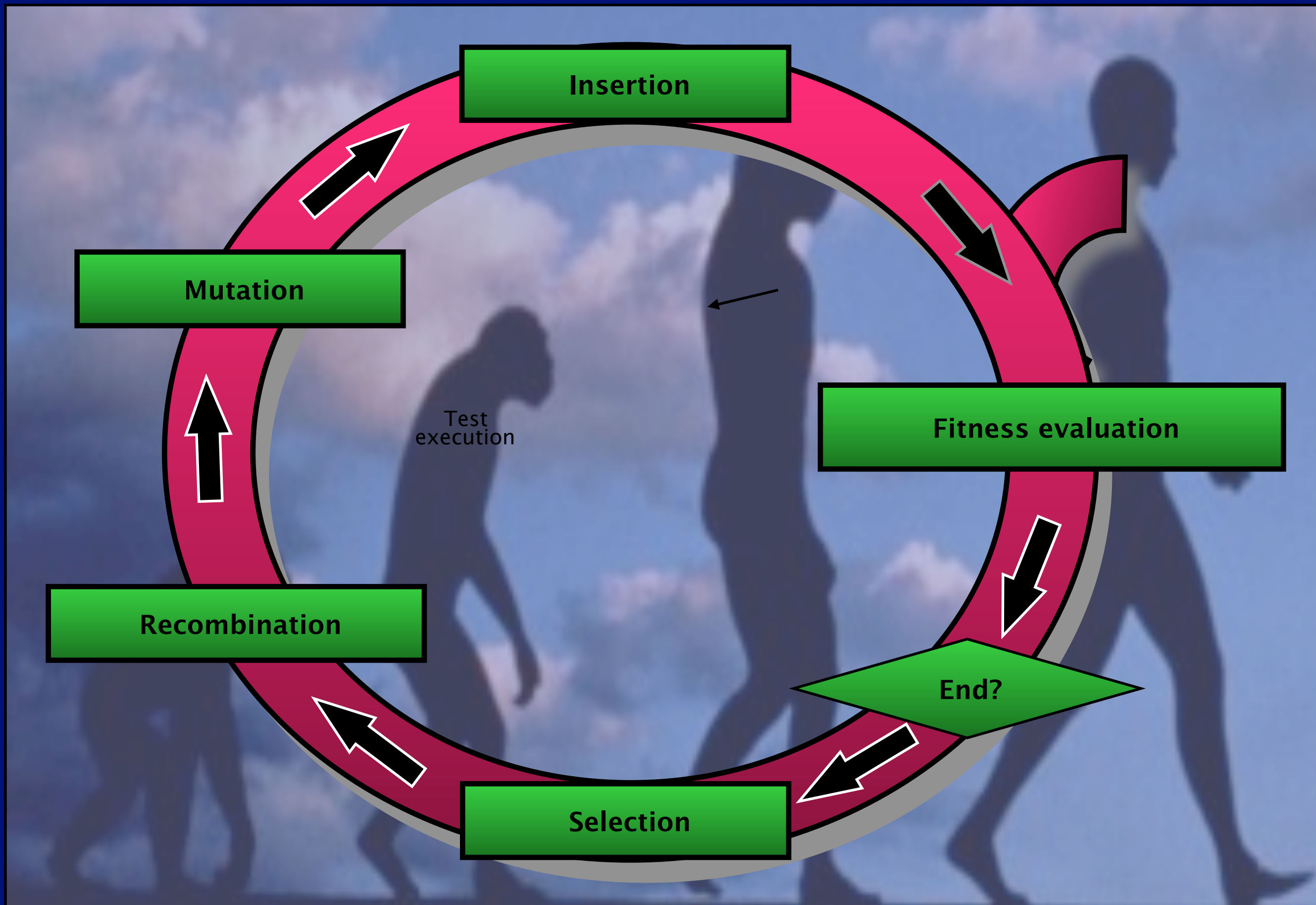
Case b :-

Decrease probe improves

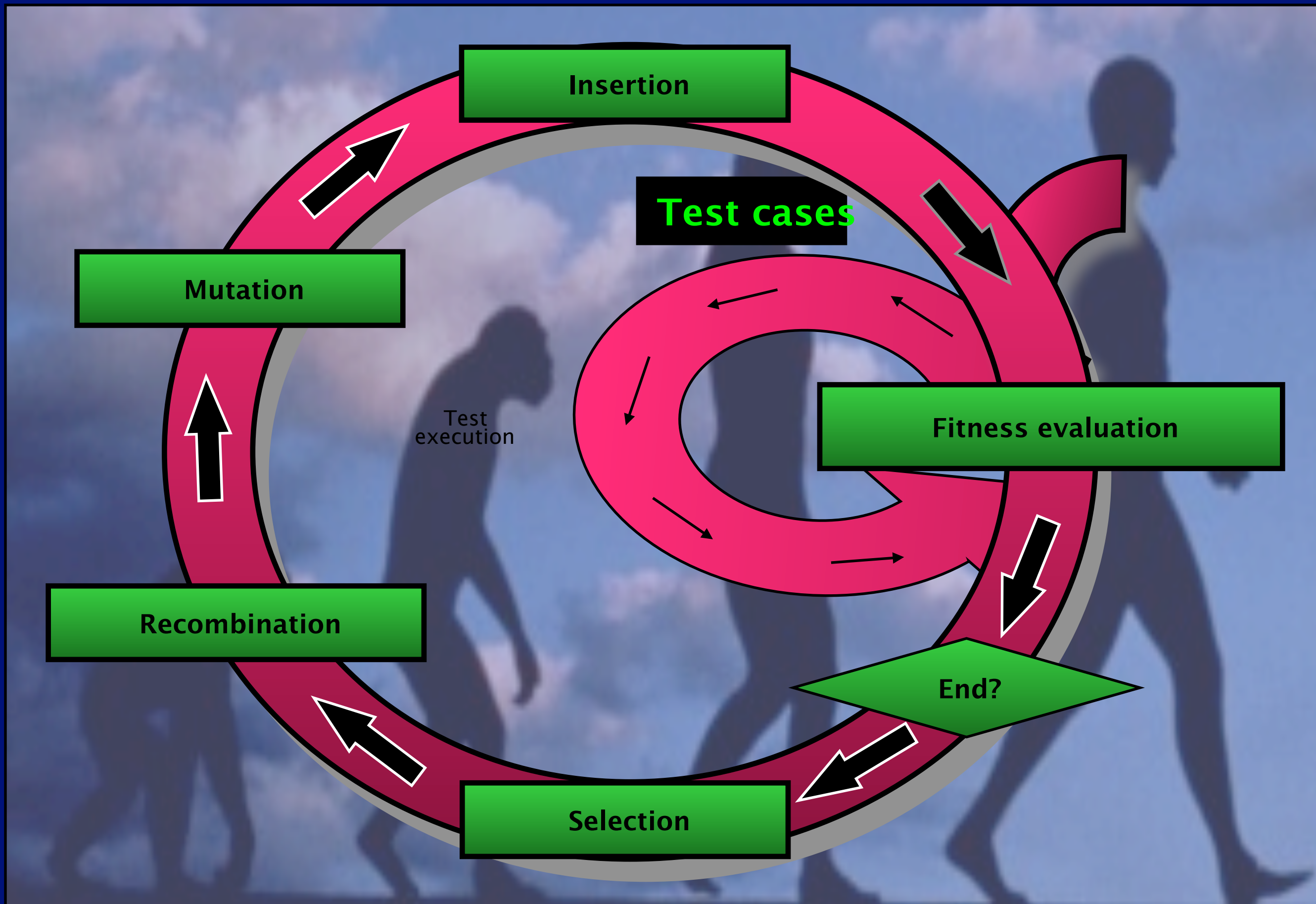
So accelerate until b=0

Case c :-

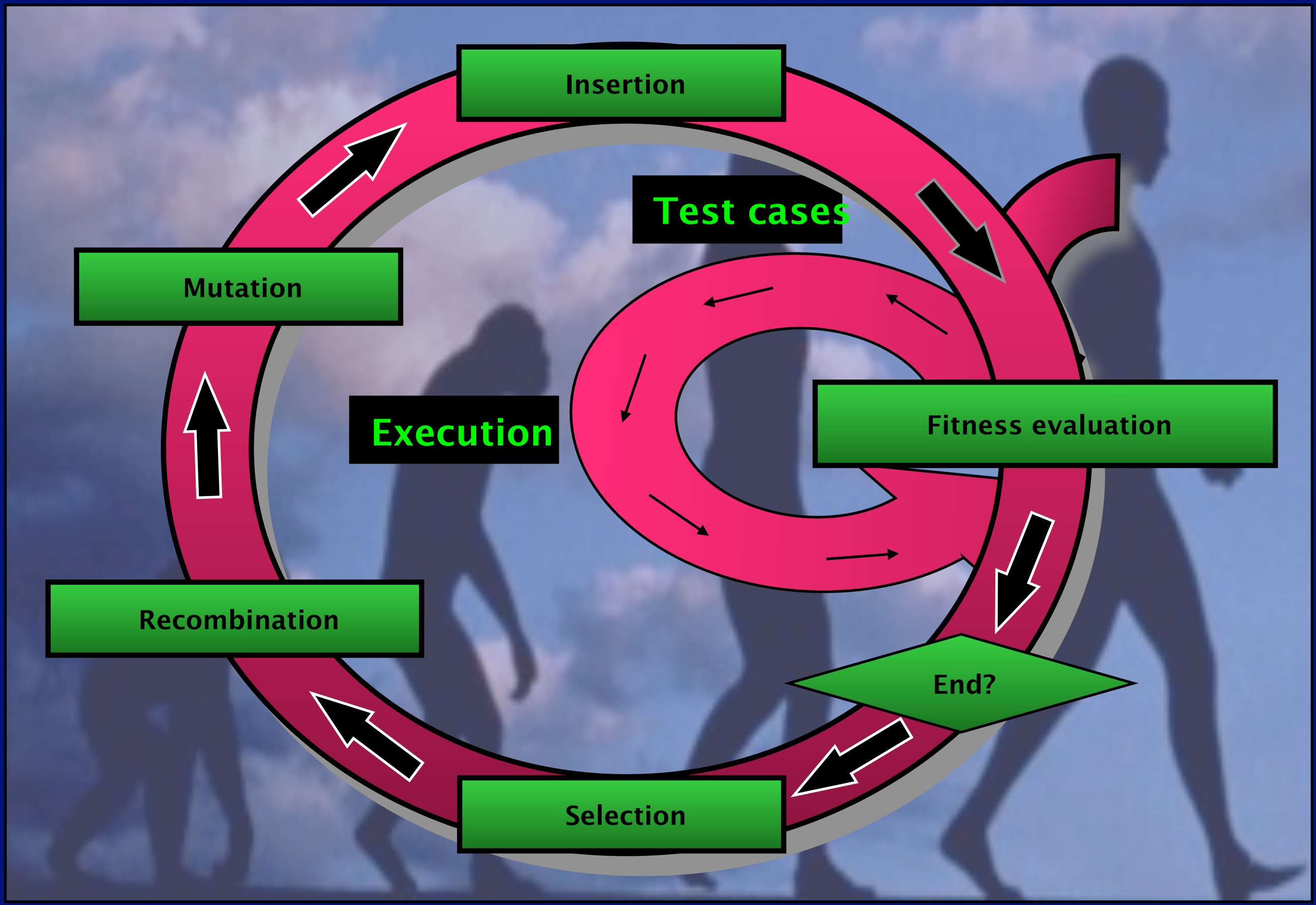
Evolutionary Algorithms



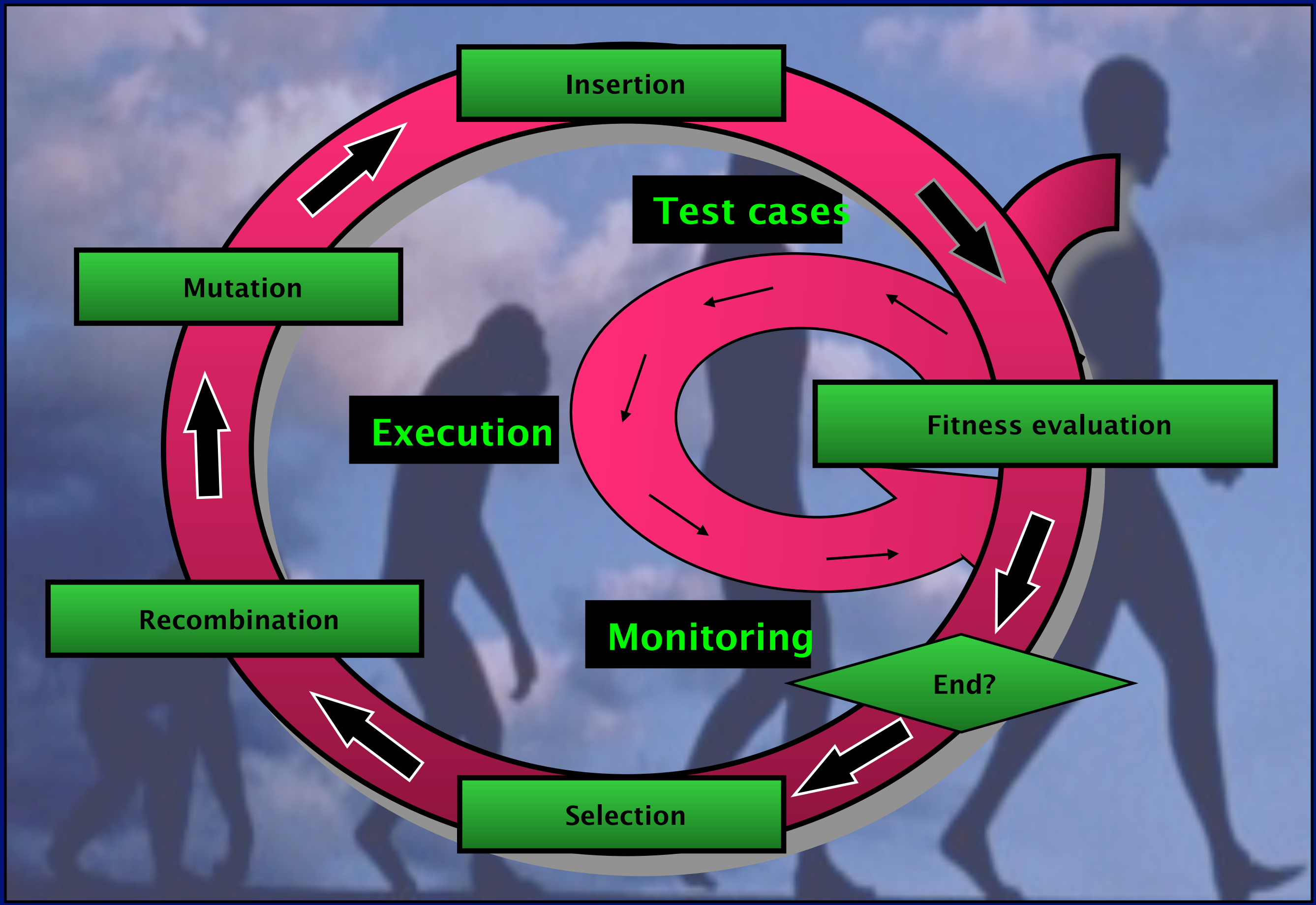
Evolutionary Testing



Evolutionary Testing



Evolutionary Testing



How mating makes life easier

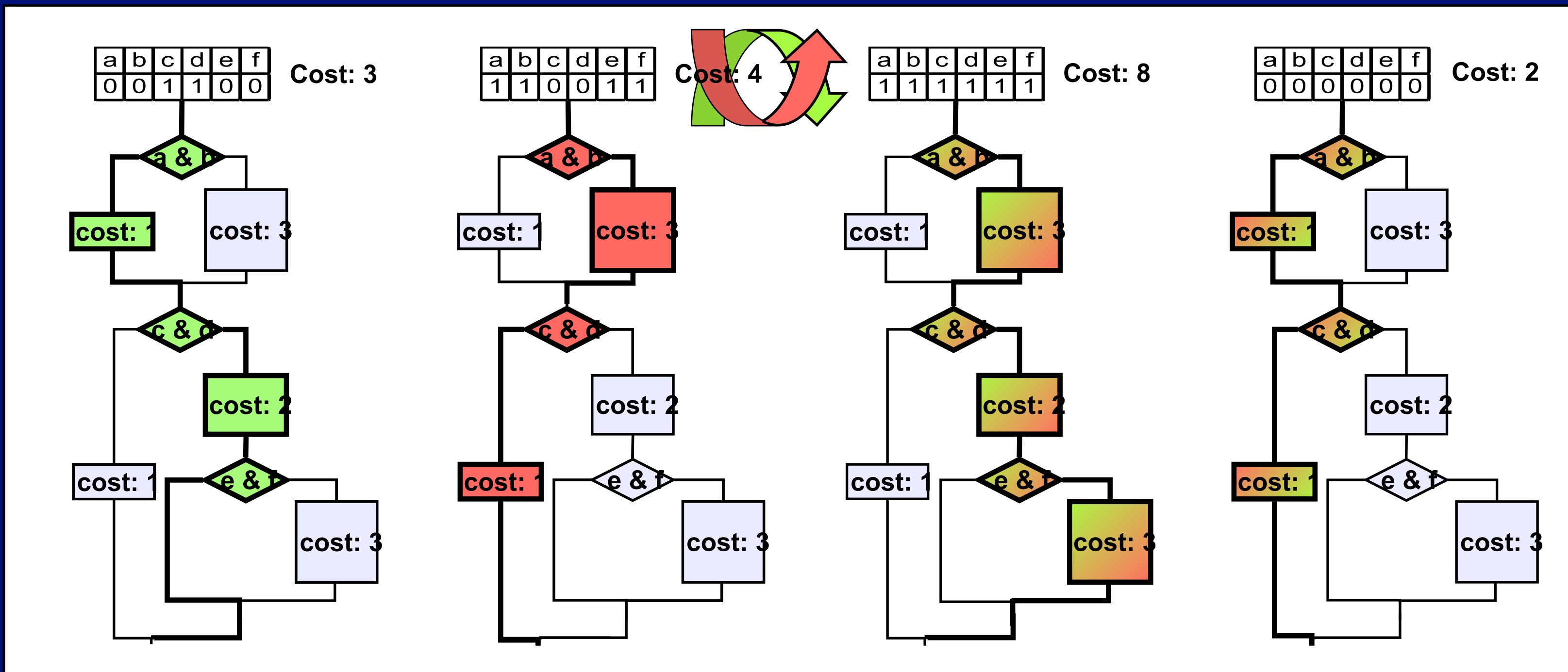
How mating makes life easier

Mating is really very much an analogy

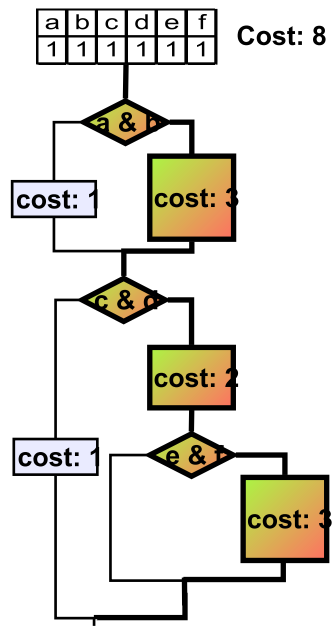
The important property is ***crossover***

How mating makes ***life*** easier

How mating makes *life* easier

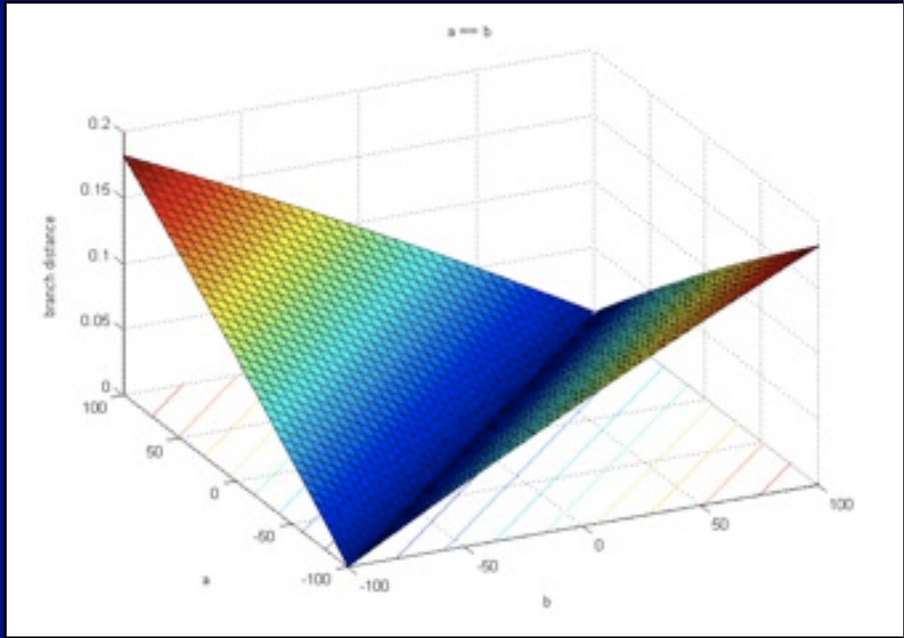


How mating makes *life* easier



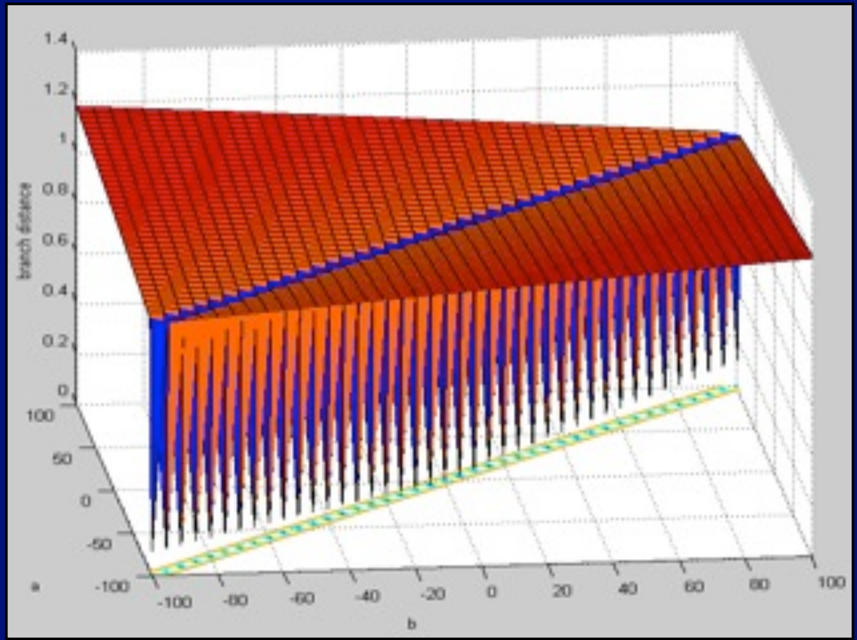
Fitness Landscape

```
void MrNiceGuy(double a, double b)
{
     if (a == b)
     {
          // target 1
          double c = b + 1;

          if (c == 0)
          {
               // target 2
. . .
```

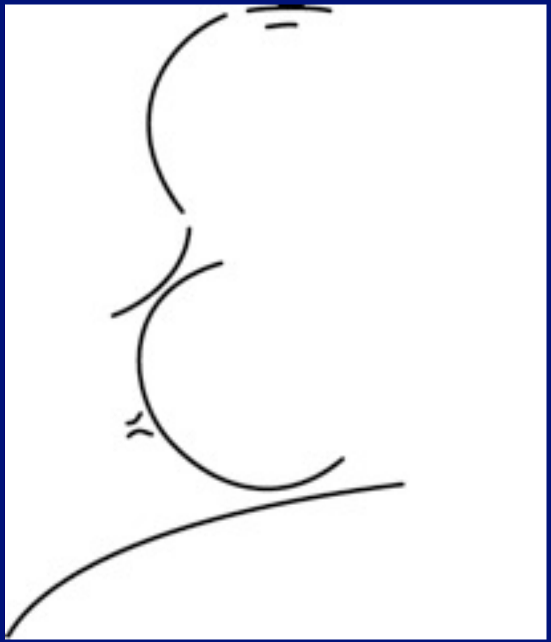


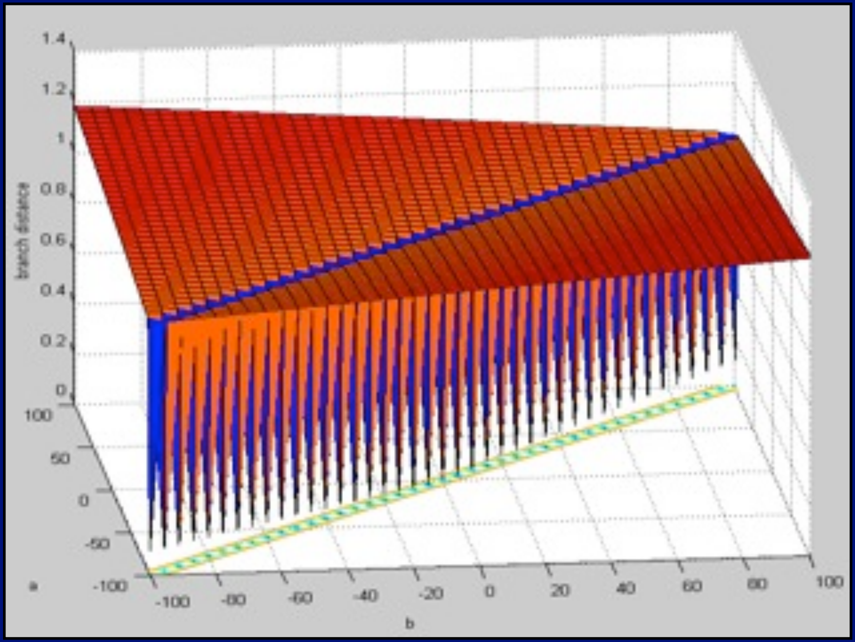
Hitchcock Fitness Landscape

```
void alfred(double a, double b)
{
    if (a == b)
    {
        // target 1
        double c = b + 1;

        if (c == 0)
        {
            // target 2
. . .
```

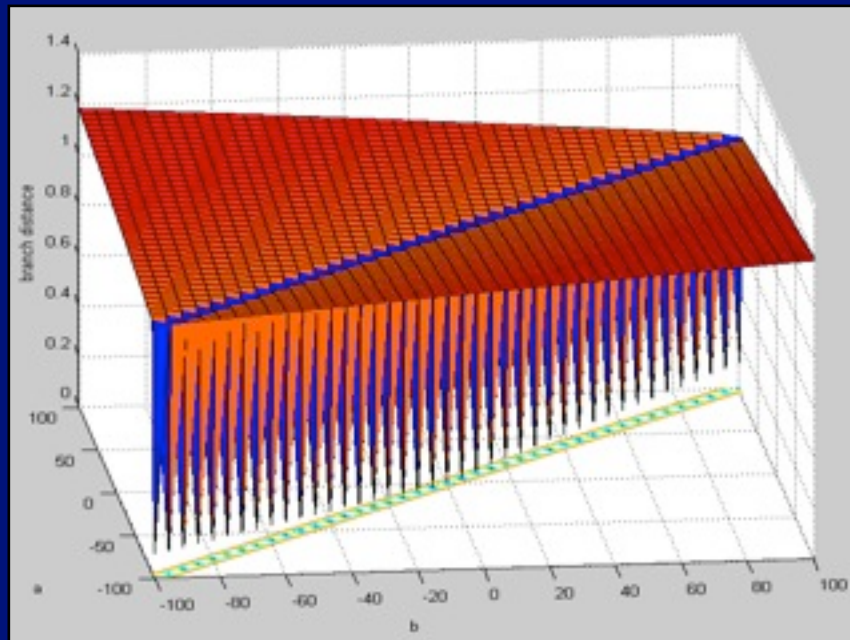


Hitchcock Fitness Landscape





Hitchcock Fitness Landscape



But …

When does it work

Why does it work (when it does)?

How does it compare to local search?

Schemas

```
Pop[1] = 010111001010100110010010001
Pop[2] = 101001001010101011111000000
Pop[3] = 010100101010101000001010110
Pop[4] = 010111101010101111110101001
```

...

Schemas

```
Pop[1] = 01011100101010011001001000 1
Pop[2] = 101001001010101011111000000
Pop[3] = 010100101010101000001010110
Pop[4] = 010111101010101111110101001
```

...

Schemas

```
Pop[1] = 010111001010100110010010001
Pop[2] = 101001001010101011111000000
Pop[3] = 010100101010101000001010110
Pop[4] = 010111101010101111110101001
...
```

Schemas

```
Pop[1] = 0101110010101001100100100001
Pop[2] = 1010010010101010111111000000
Pop[3] = 0101001010101010000001010110
Pop[4] = 0101111010101011111110101001
```

...

Schemas

$$\overline{f}(\mathbf{h}, K) = \frac{1}{|\{x \mid x \in \mathbf{h} \wedge x \in K\}|} \sum_{x \in \mathbf{h} \wedge x \in K} f(x)$$

$$N(\mathbf{h}, g+1) \geq N(\mathbf{h}, g) \frac{\overline{f}(\mathbf{h}, P(g))}{\frac{1}{M} \sum_{x \in P(g)} f(x)}$$

$$N(\mathbf{h}, g+1) \geq N(\mathbf{h}, g) \frac{\overline{f}(\mathbf{h}, P(g))}{\frac{1}{M} \sum_{x \in P(g)} f(x)} (1 - p_c \frac{\delta(\mathbf{h})}{\lambda - 1} - p_m\, o(\mathbf{h}))$$

Schemas

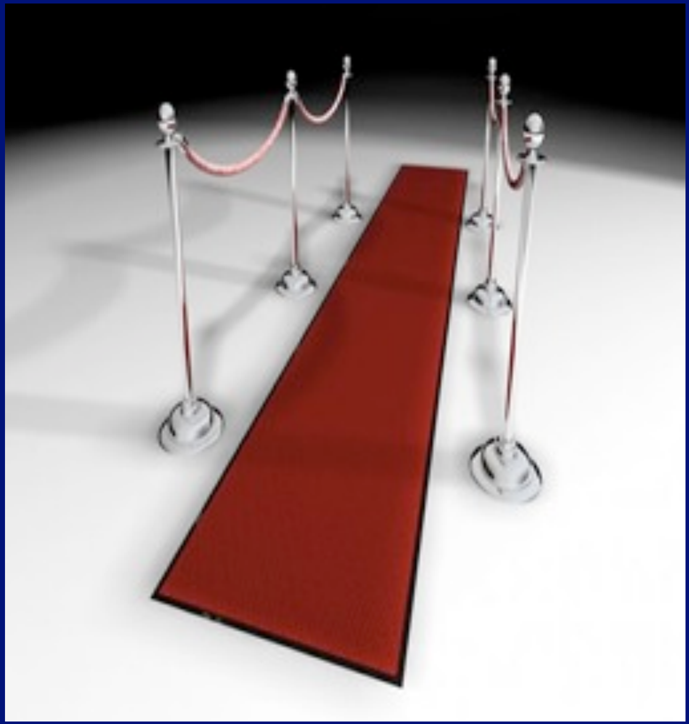
$$\overline{f}(\mathbf{h}, K) = \frac{1}{|\{x \mid x \in \mathbf{h} \wedge x \in K\}|} \sum_{x \in \mathbf{h} \wedge x \in K} f(x)$$

$$N(\mathbf{h}, g+1) \geq N(\mathbf{h}, g) \frac{\overline{f}(\mathbf{h}, P(g))}{\frac{1}{M} \sum_{x \in P(g)} f(x)}$$

$$N(\mathbf{h}, g+1) \geq N(\mathbf{h}, g) \frac{\overline{f}(\mathbf{h}, P(g))}{\frac{1}{M} \sum_{x \in P(g)} f(x)} \left(1 - p_c \frac{\delta(\mathbf{h})}{\lambda - 1} - p_m\, o(\mathbf{h})\right)$$

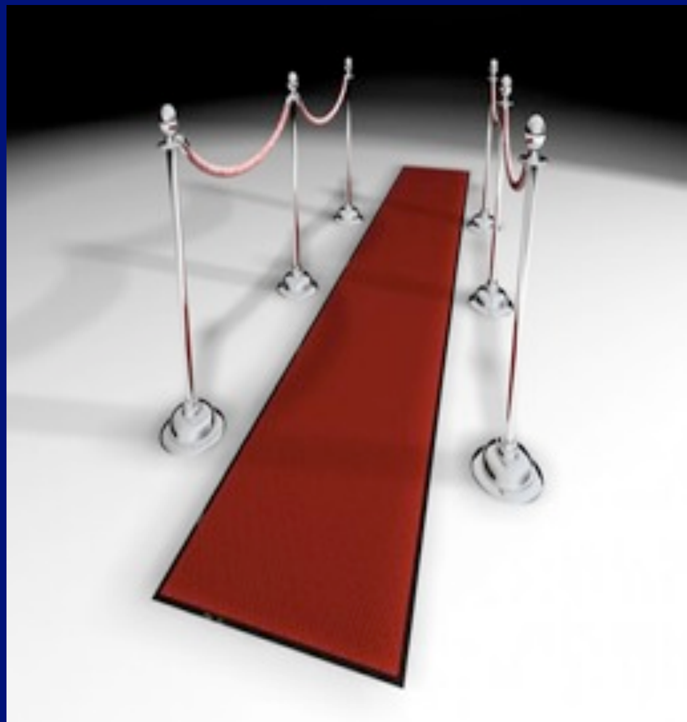
The Royal Road

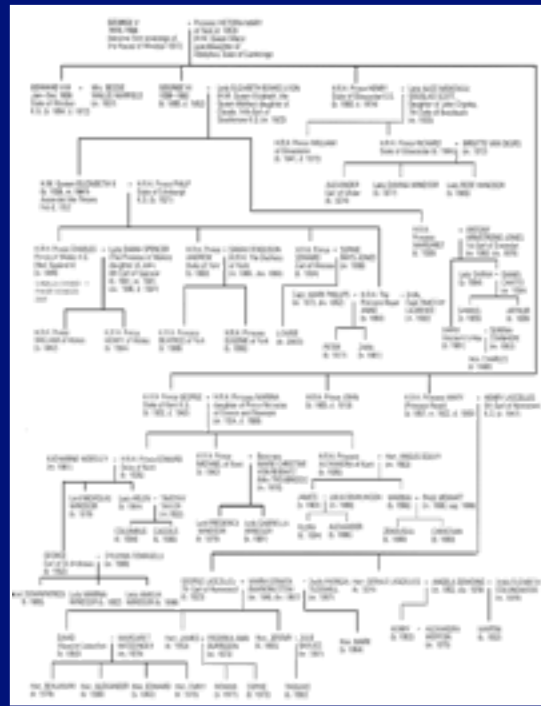
The Royal Road



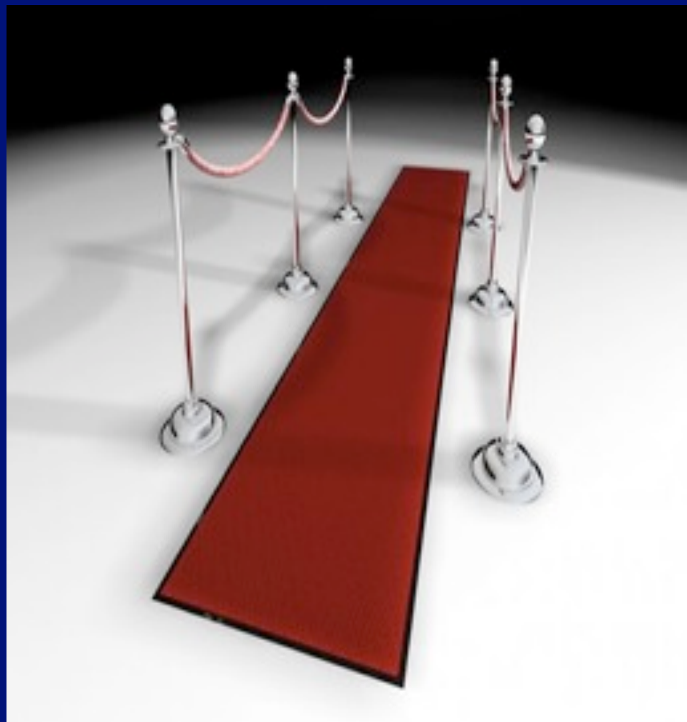
The Royal Road

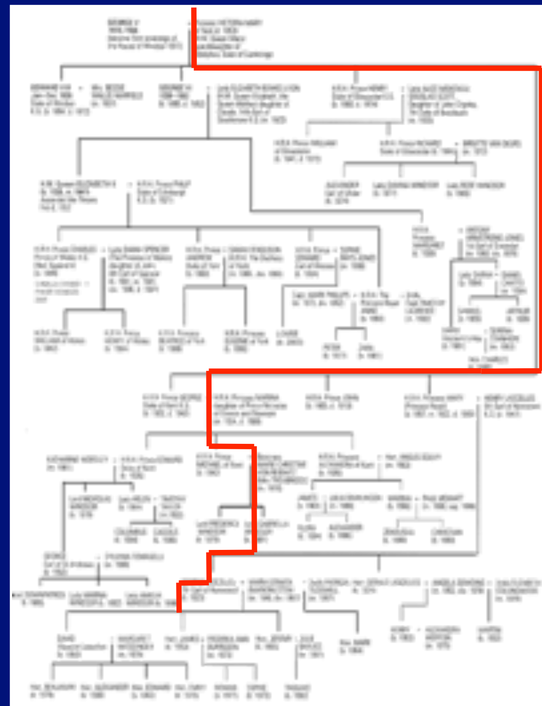
The Royal Road



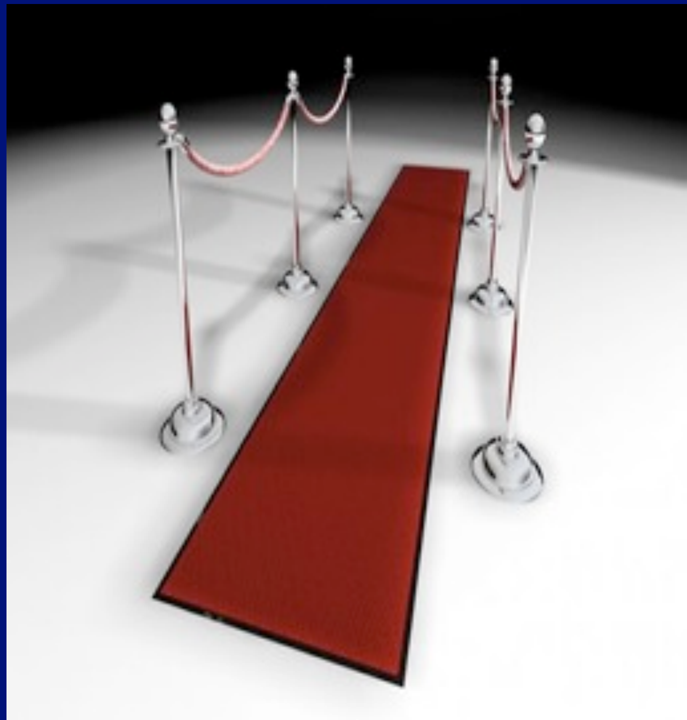


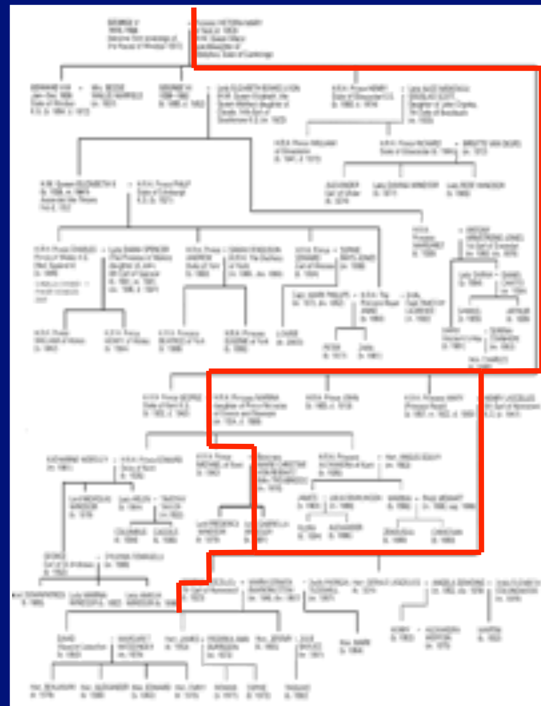
The Royal Road





The Royal Road





The Royal Road

```
S1:  1111****************************
S2:  ****1111************************
S3:  ********1111********************
S4:  ************1111****************
S5:  ****************1111************
S6:  ********************1111********
S7:  ************************1111****
S8:  ****************************1111
S9:  11111111************************
S10: ********11111111****************
S11: ****************11111111********
S12: ************************11111111
S13: 1111111111111111****************
S14: ****************1111111111111111
S15: 11111111111111111111111111111111
```

The Royal Road

```
S1:  1111****************************
S2:  ****1111************************
S3:  ********1111********************
S4:  ************1111****************
S5:  ****************1111************
S6:  ********************1111********
S7:  ************************1111****
S8:  ****************************1111
S9:  11111111************************
S10: ********11111111****************
S11: ****************11111111********
S12: ************************11111111
S13: 1111111111111111****************
S14: ****************1111111111111111
S15: 11111111111111111111111111111111
```

The Royal Road

```
S1:  1111****************************
S2:  ****1111************************
S3:  ********1111********************
S4:  ************1111****************
S5:  ****************1111************
S6:  ********************1111********
S7:  ************************1111****
S8:  ****************************1111
S9:  11111111************************
S10: ********11111111****************
S11: ****************11111111********
S12: ************************11111111
S13: 1111111111111111****************
S14: ****************1111111111111111
S15: 11111111111111111111111111111111
```

The Royal Road

```
S1:  1111****************************
S2:  ****1111************************
S3:  ********1111********************
S4:  ************1111****************
S5:  ****************1111************
S6:  ********************1111********
S7:  ************************1111****
S8:  ****************************1111
S9:  11111111************************
S10: ********11111111****************
S11: ****************11111111********
S12: ************************11111111
S13: 1111111111111111****************
S14: ****************1111111111111111
S15: 11111111111111111111111111111111
```

The Royal Road

```
S1:  1111****************************
S2:  ****1111************************
S3:  ********1111********************
S4:  ************1111****************
S5:  ****************1111************
S6:  ********************1111********
S7:  ************************1111****
S8:  ****************************1111
S9:  11111111************************
S10: ********11111111****************
S11: ****************11111111********
S12: ************************11111111
S13: 1111111111111111****************
S14: ****************1111111111111111
S15: 11111111111111111111111111111111
```

The Royal Road

```
S1:  1111****************************
S2:  ****1111************************
S3:  ********1111********************
S4:  ************1111****************
S5:  ****************1111************
S6:  ********************1111********
S7:  ************************1111****
S8:  ****************************1111
S9:  11111111************************
S10: ********11111111****************
S11: ****************11111111********
S12: ************************11111111
S13: 1111111111111111****************
S14: ****************1111111111111111
S15: 11111111111111111111111111111111
```

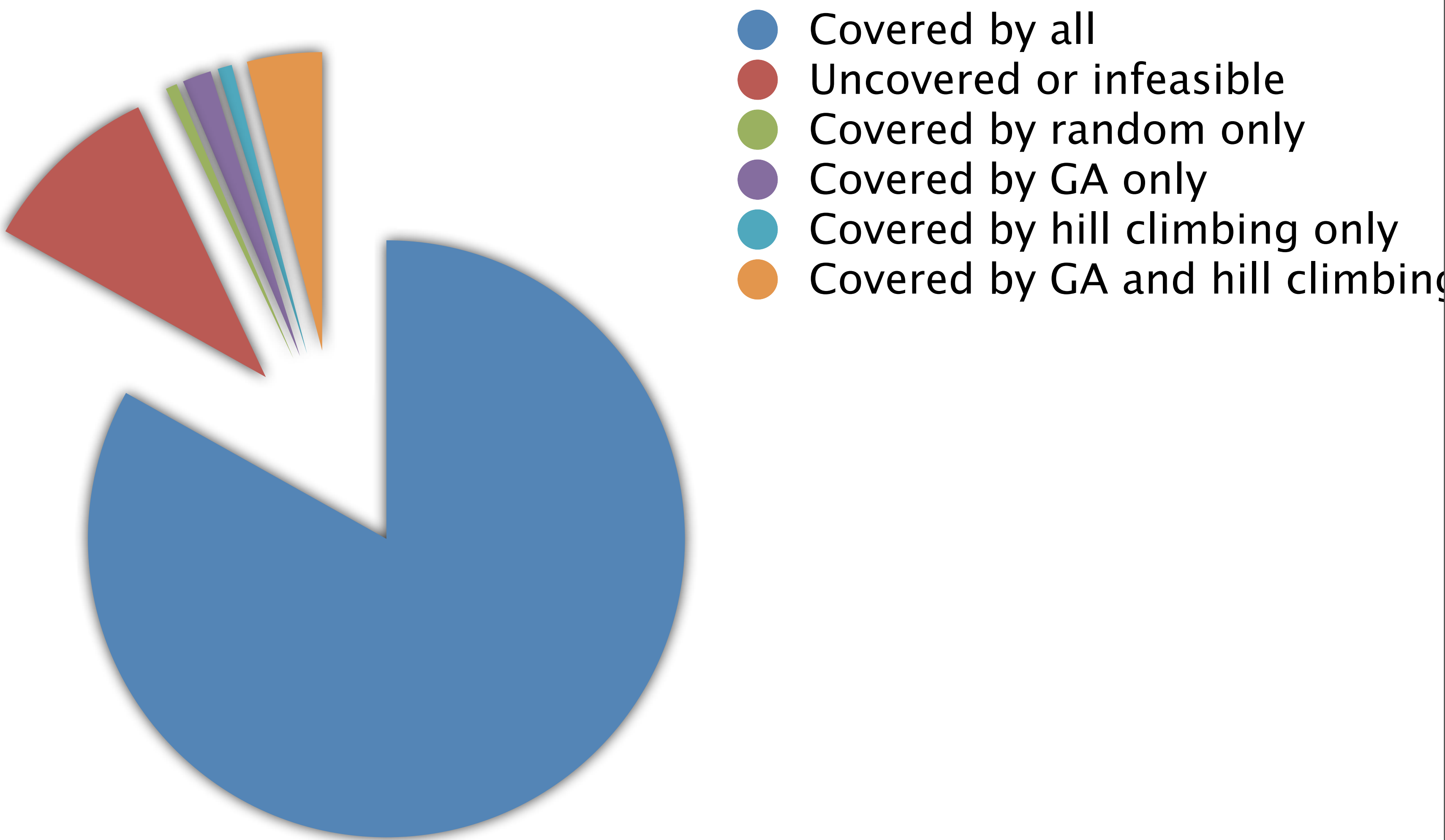
Subjects

Bibclean	open source BibTeX pretty printer
Eurocheck	open source € serial number validation
Gimp	open source image manipulation
Spice	analogue circuit simulator
Tiff	TIFF library for image manipulation
Space	ever heard of this one?

Experimental set up

Fitness evaluations	100,000
Executions	30
Same seeds	for statistical testing

Overall Results



Results

From 640 branches in the six subjects

- 10 branches
  - for which Evolutionary Testing was successful but a simple Hill Climb search was not
- 5 branches
  - for which Hill Climbing was successful but Evolutionary Testing was not
- 26 branches
  - Comparability

Royal Roads?

8 branches

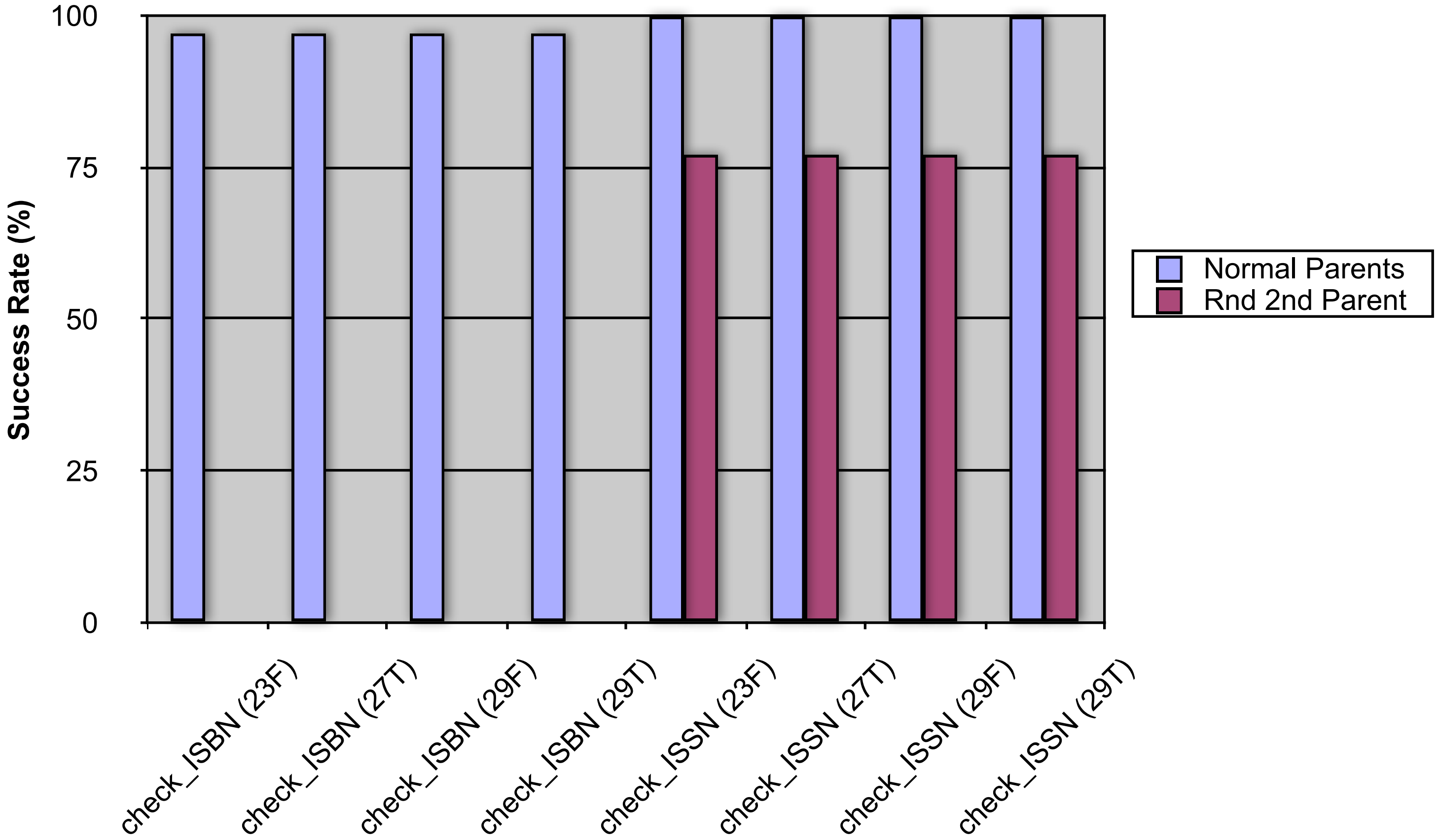
Evolutionary Testing succeeds where Hill Climbing fails:

Branches of the bibclean test object

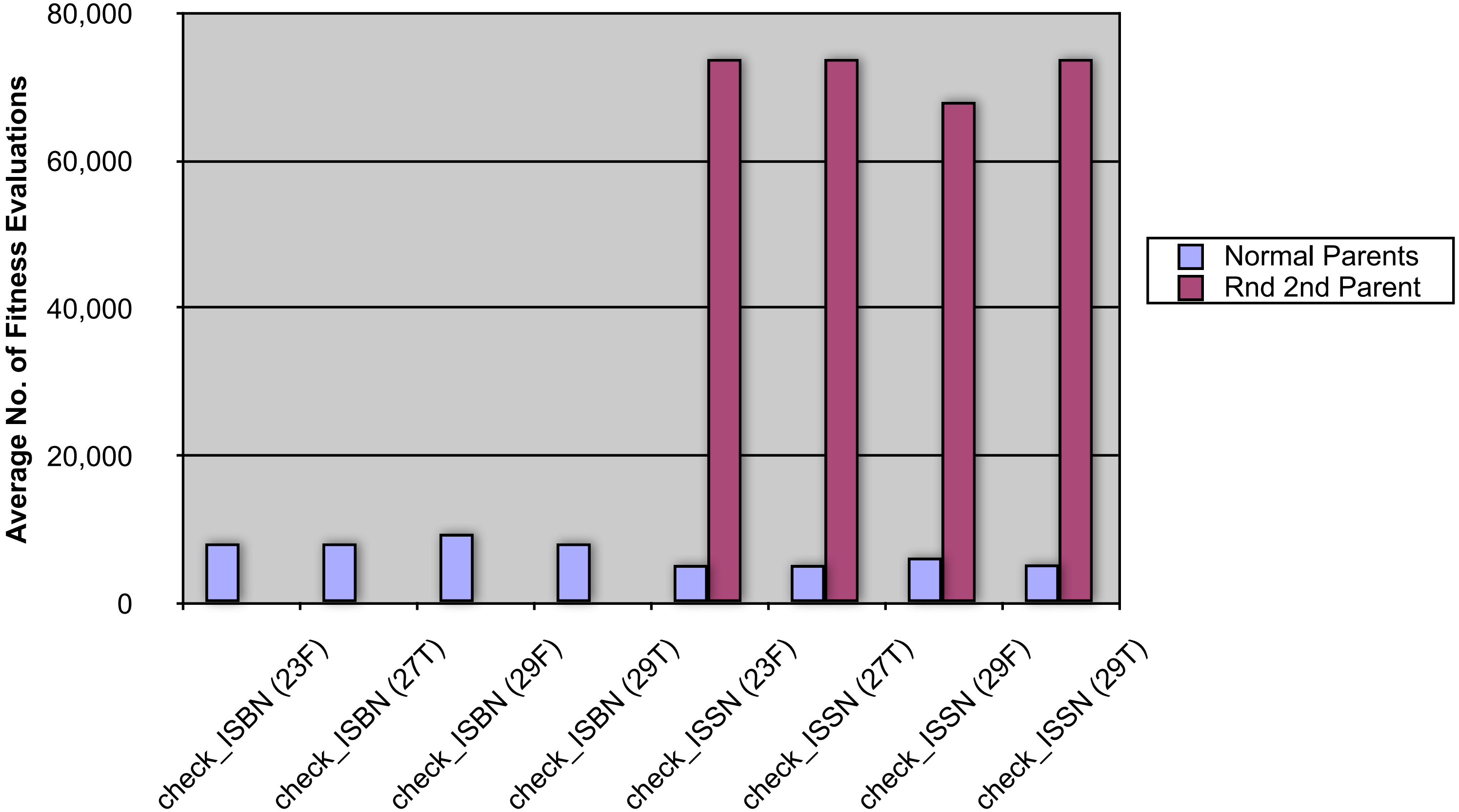
String check for a valid ISBN/ISSN

At least 10 digits

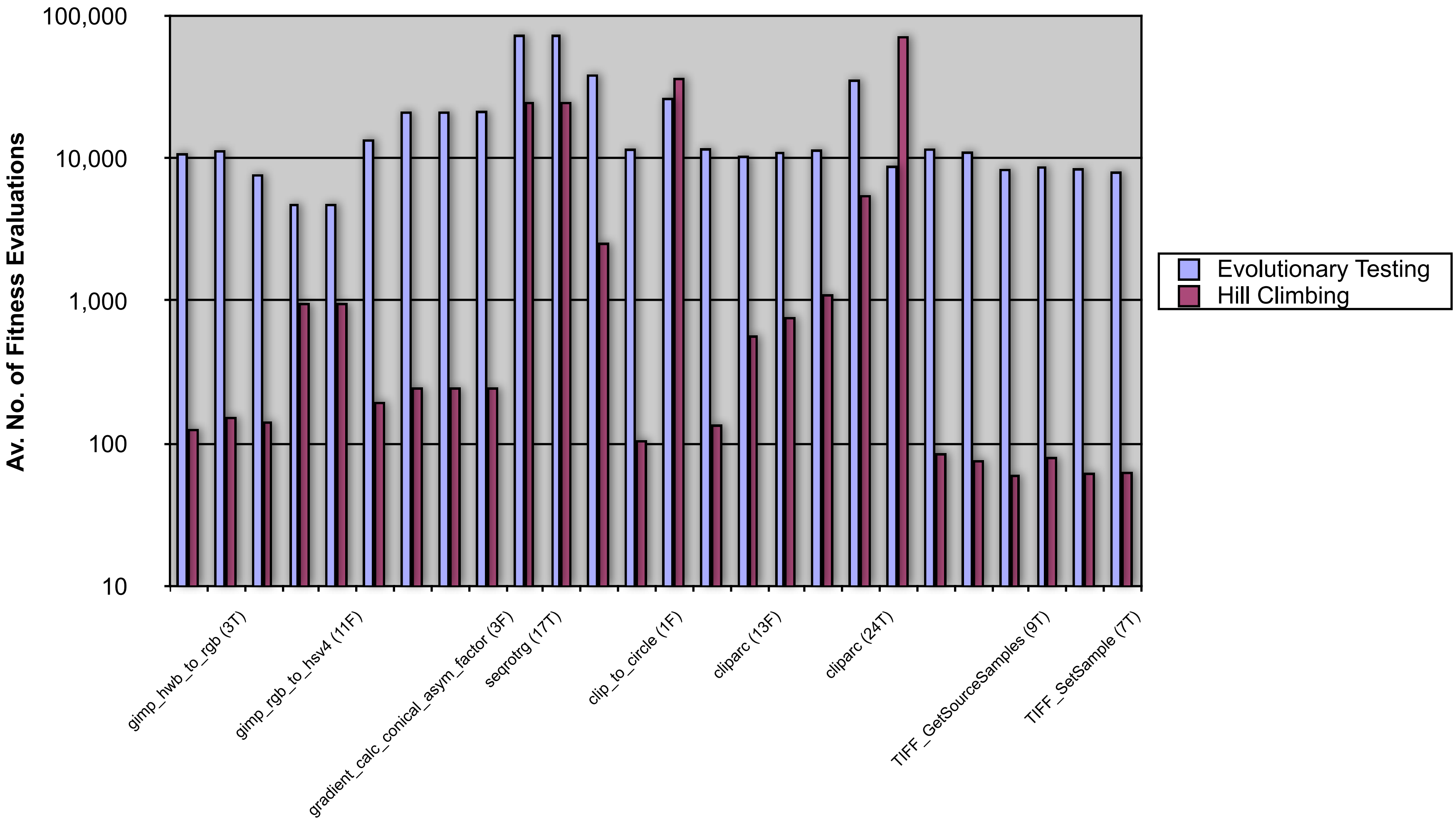
Headless chicken test: success rate



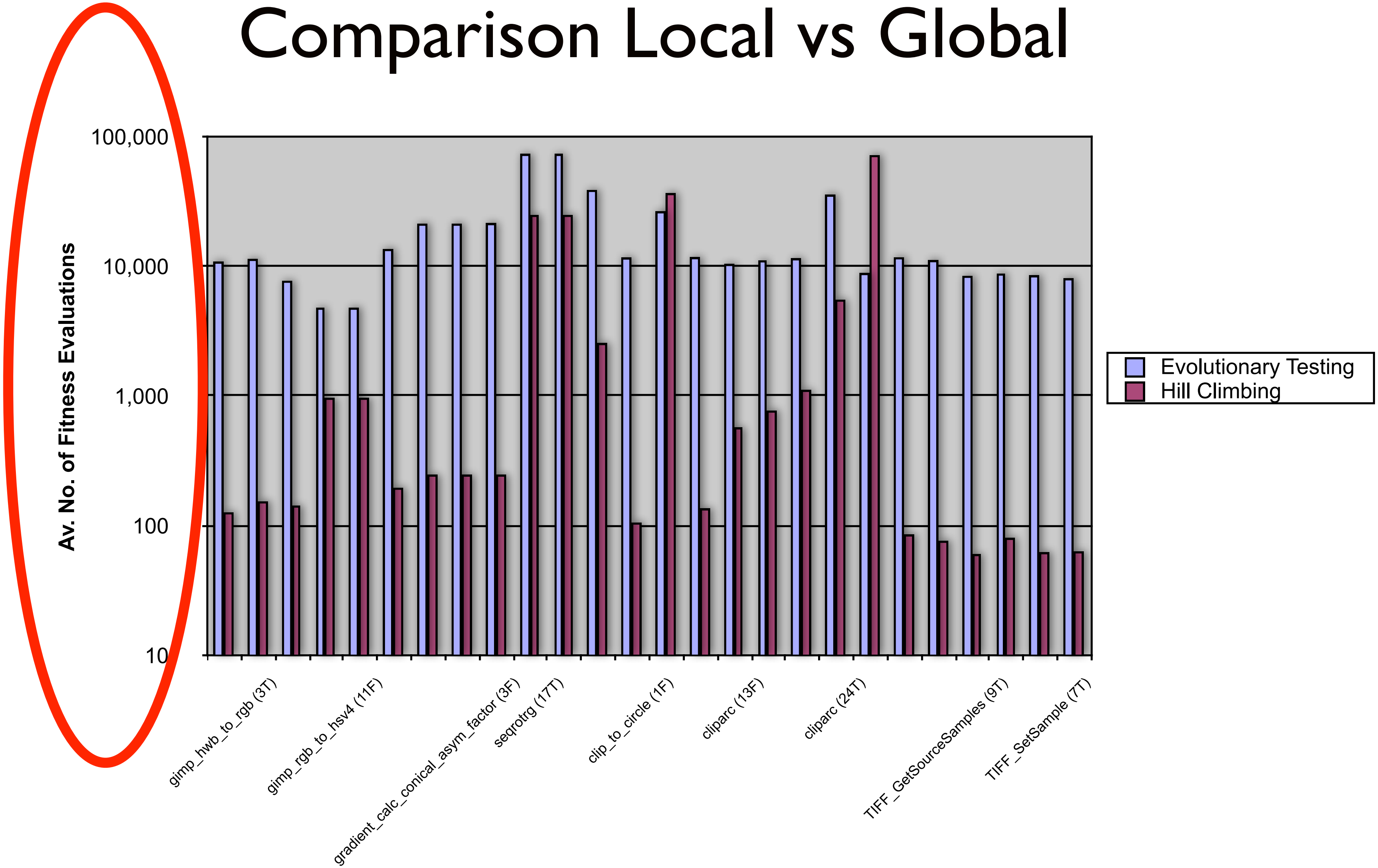
Headless chicken test: test effort



Comparison Local vs Global



Comparison Local vs Global



HC outperforms GA

HC is fast, easy and effective

In 24 of the 26 comparable cases it beats GA

Average speed up is approximately a factor of 20

The results were statistically significant (paired *t* test)

Conclusions

GA does perform well for Royal Road Functions
… and this is because of the cross over operator

But how many real programs have royal roads?

For those which don't HC is comfortably faster
… by an order of magnitude
… evolution strategies may outperform GA for RR

Of course random covers most branches
… but only the easy ones

Future work

Memetic algorithms

Evolution strategies

Multi objective test data generation

Study of SBT and DART

Other GA theories