

Reducing Bloat in GP with Multiple Objectives

Stefan Bleuler

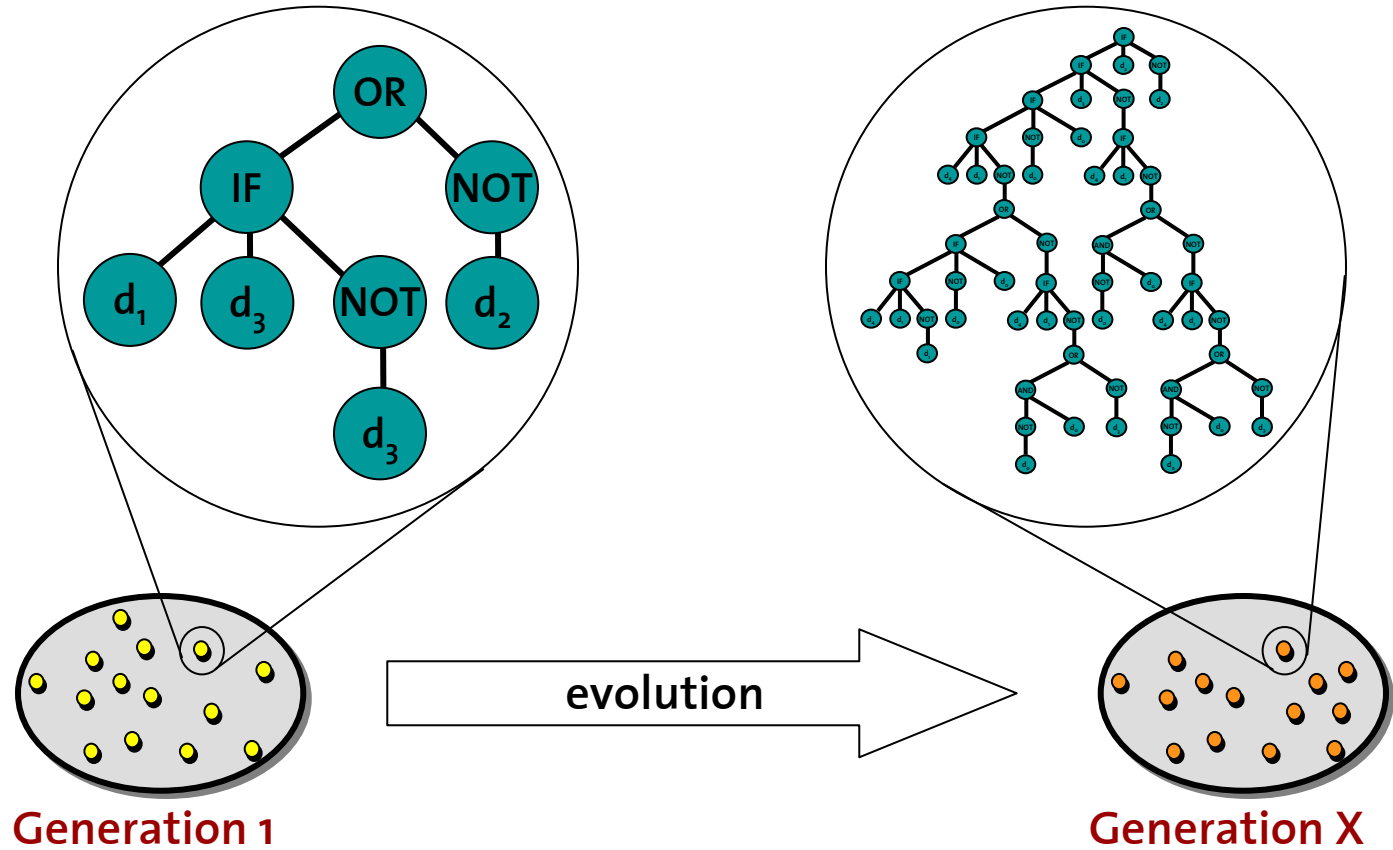
Johannes Bader, Martin Brack and Eckart Zitzler

PPSN Workshop, 2006



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Problem: Bloat in Genetic Programming



Bloat: average tree size increases rapidly

Causes and Effects of Large Trees

Possible Causes of Bloat

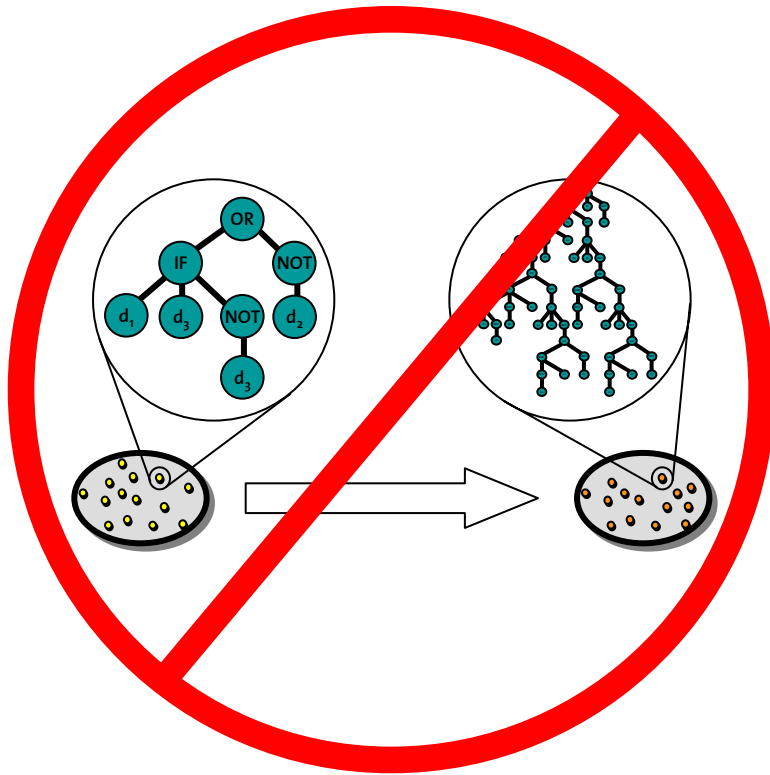
- Introns: useless code to protect from disruptive crossover
- Distribution: there are just more large solutions than small ones

Main Effects of Large Trees

- high resource usage during optimization
- high runtime of evolved solution
- ineffective crossover
- small solutions generalize better
- + solution might be large
- + enough genetic material

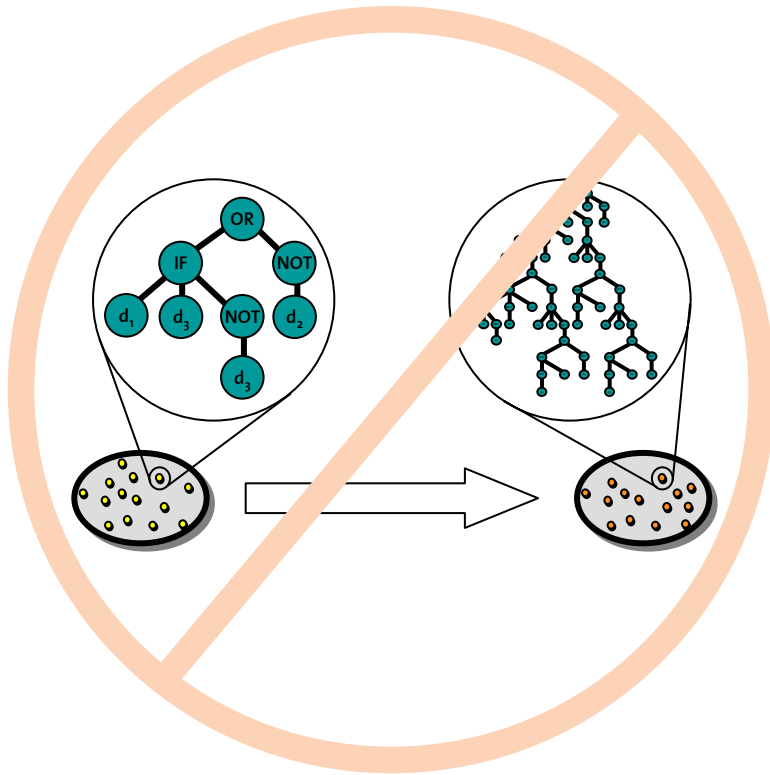
→ How to reduce bloat?

Overview



- 1 Traditional Methods
- 2 Multiobjective Approaches
- 3 Why are they effective?
A preliminary answer.

Overview



- 1** Traditional Methods
- 2** Multiobjective Approaches
- 3** Why are they effective?
A preliminary answer.

Traditional Methods

$$Fitness_i = f(Error_i, Size_i)$$

Standard GP with limited tree depth (Koza 92):

$$\begin{aligned} Fitness_i &= Error_i && \text{if } Size_i \leq MaxSize \\ Fitness_i &= \infty && \text{if } Size_i > MaxSize \end{aligned}$$

- commonly used
 - bloating up to limit
 - constraint method
 - extension: dynamic limit (Silva, Costa, 2004)
-

Constant Parsimony pressure (Koza 92):

$$Fitness_i = Error_i + \alpha \cdot Size_i$$

- manual setting of α
- weighted sum approach

Traditional Methods II

$$Fitness_i = f(Error_i, Size_i)$$

Two Stage (Kalganova, Miller 99):

$$a) Fitness_i = Error_i + 1 \quad \text{if } Error_i > \varepsilon$$

$$b) Fitness_i = 1 - \frac{1}{Size_i} \quad \text{if } Error_i \leq \varepsilon$$

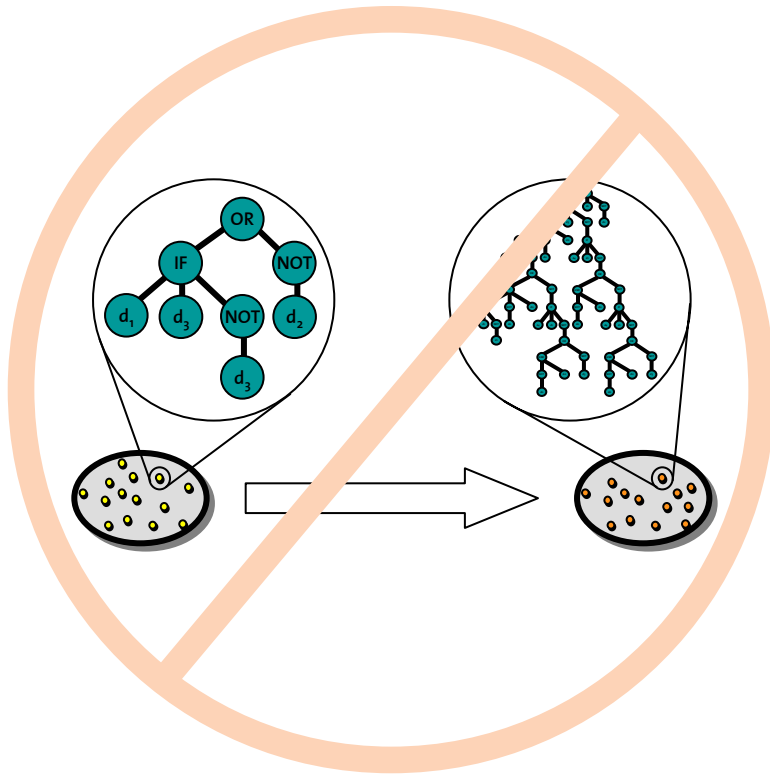
- ranking of objectives
- optimizing functionality first and size afterwards

Adaptive Parsimony pressure (Zhang, Mühlenbein 95):

$$Fitness_i = Error_i + \alpha_g \cdot Size_i$$

- automatical setting of α
- weighted sum approach
- ranking of objectives

Overview



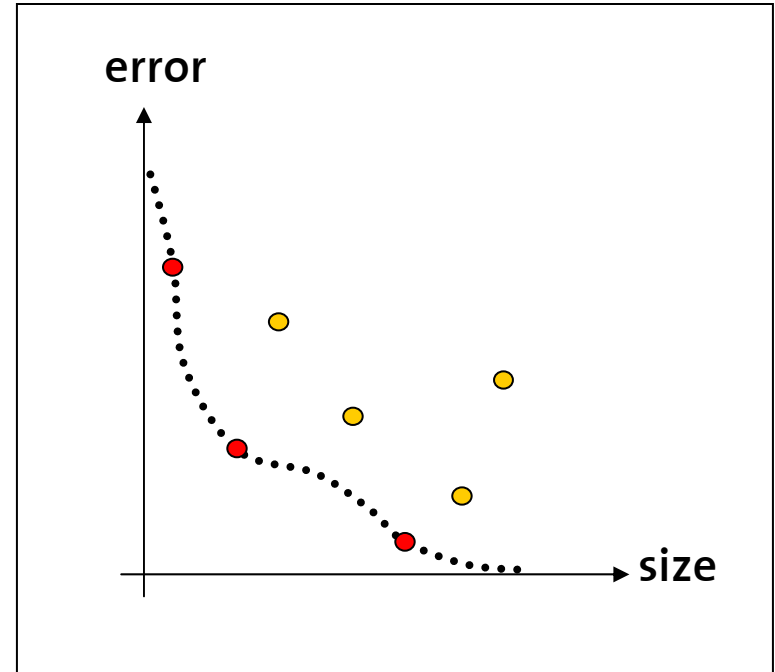
- 1 Traditional Methods
- 2 Multiobjective Approaches**
- 3 Why are they effective?
A preliminary answer.

Optimizing Size and Fitness

→ size as independent objective

Goals

- avoid choosing a parameter value
- avoid comparing size and error
- keep small individuals in the population



First Multiobjective Approaches

Non-domination Tournament (Ecart, Nemeth 2001):

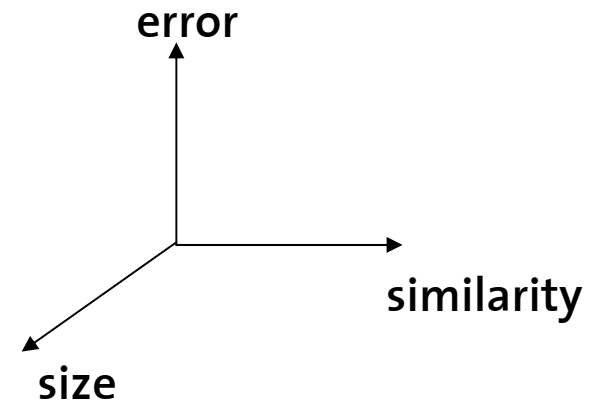
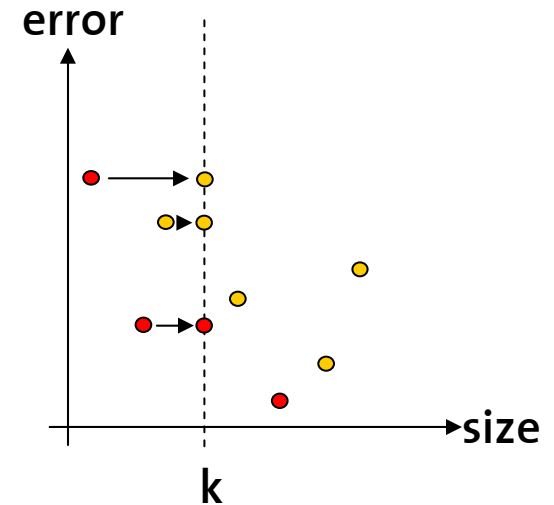
- bias domination towards fit individuals
- test problems: 3 symbolic regression, multiplexer
- results: much smaller solutions than standard GP, faster running time

FOCUS (de Jong, Watson, Pollack 2001, 2003):

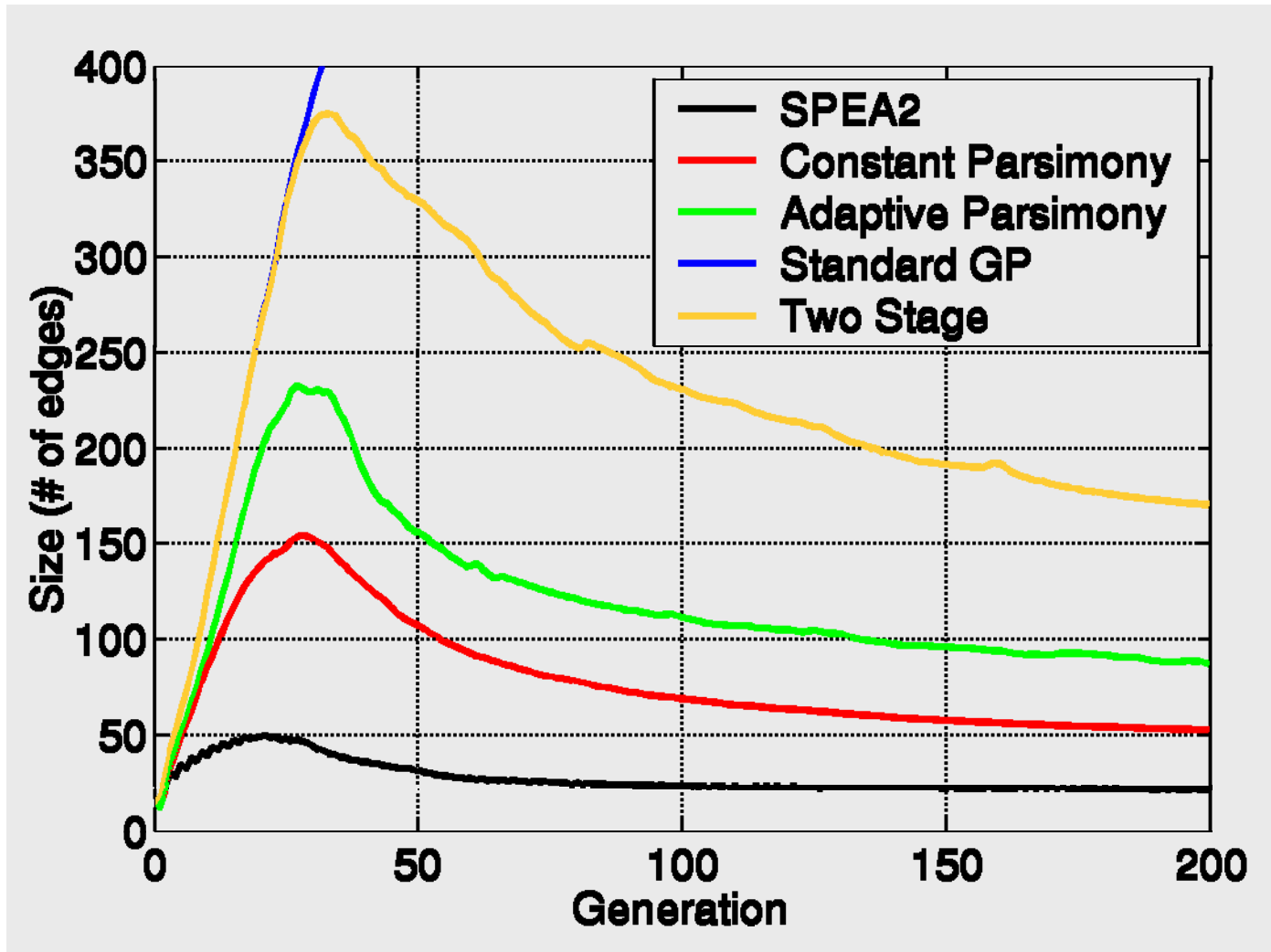
- use diversity as third objective
- diversity is measured in parameter space
- test problem: parity bit
- results: much smaller solutions than standard GP, fewer evaluations

SPEA2 (Bleuler, Brack, Thiele, Zitzler. 2001):

- maintain diversity in objective space
- test problem: parity bit

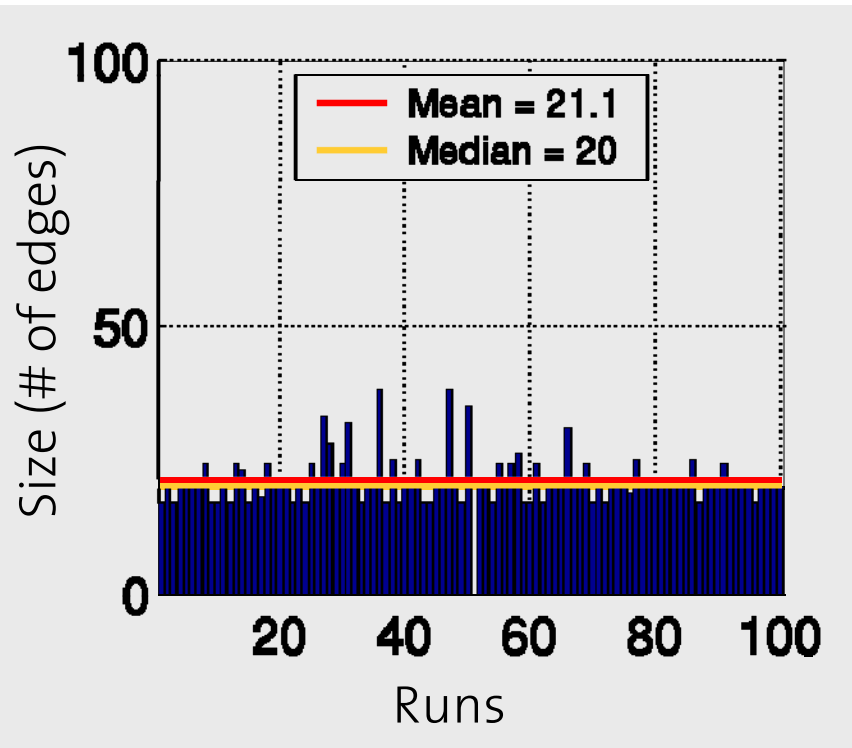


Results – Average Tree Size

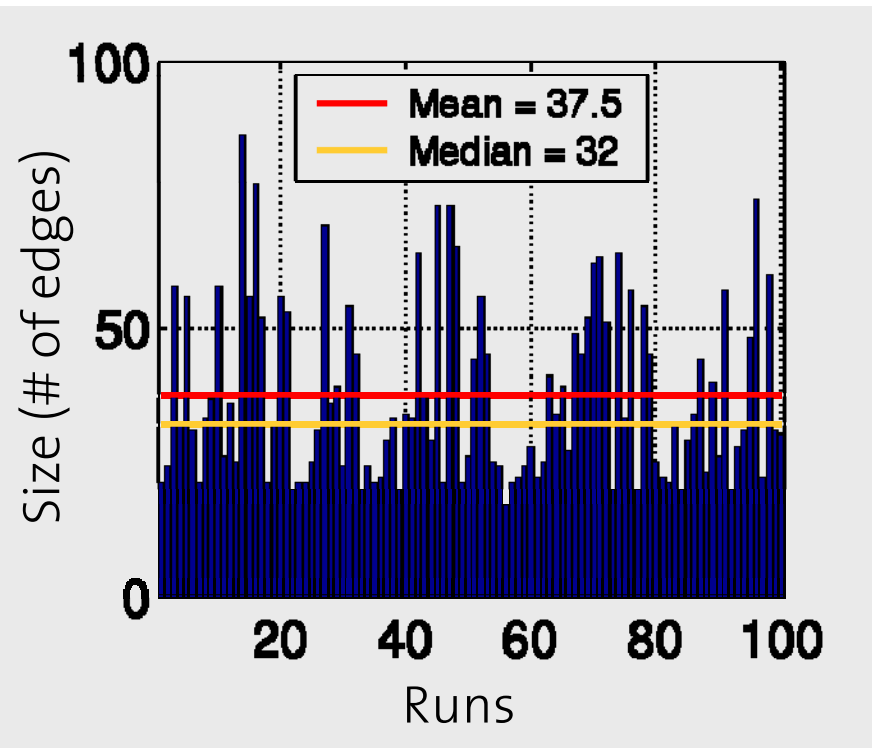


Results – Size of Correct Solutions

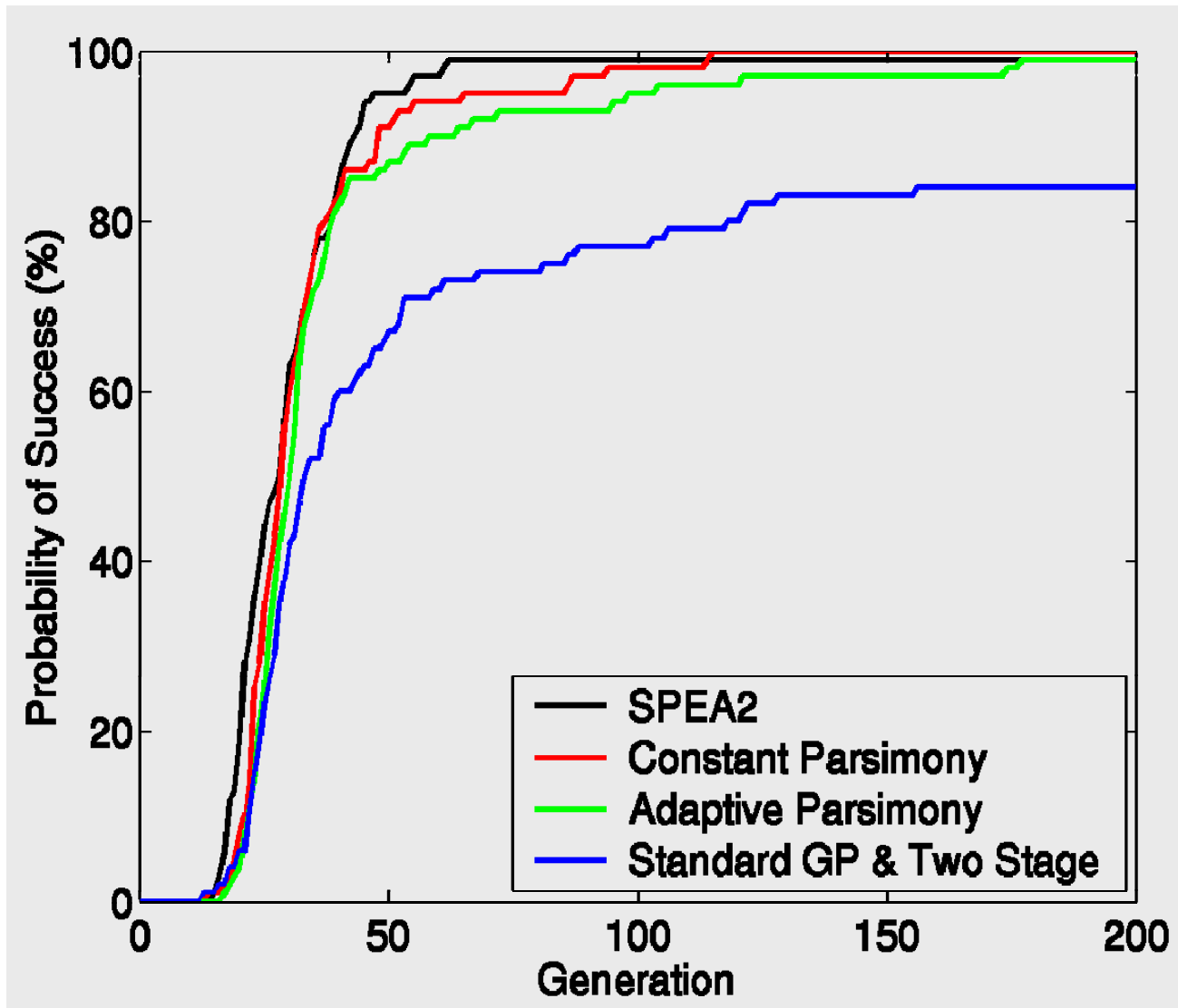
SPEA2



Constant Parsimony



Results – Probability of Success



Summary of Multiobjective Results

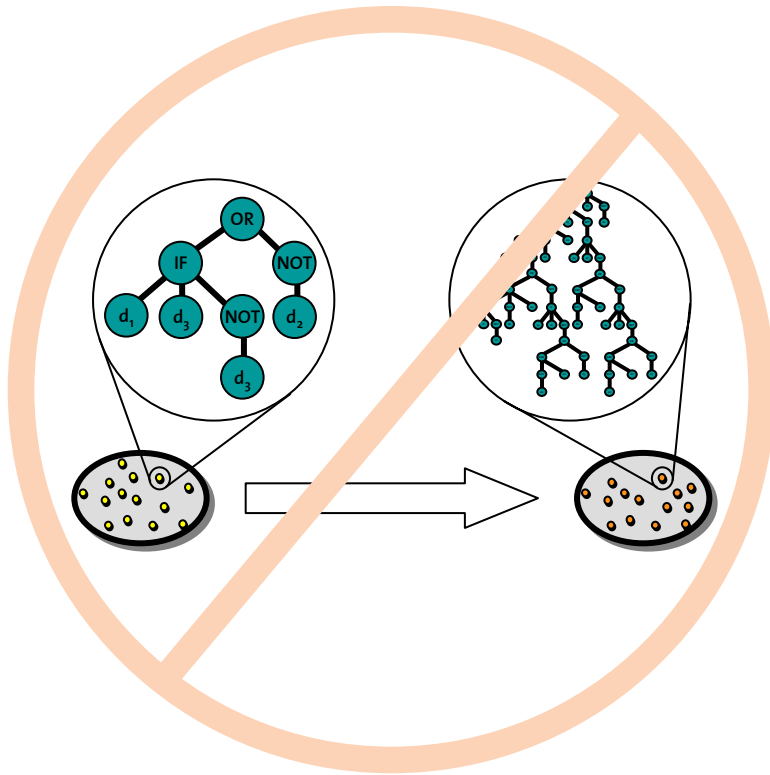
Some newer studies have confirmed good results:

- Biased Multiobjective Parsimony (Panait, Luke. 2004)
- Pareto Tournament (Kotanchek, Smits, Vladislavleva. 2006)

Key Results

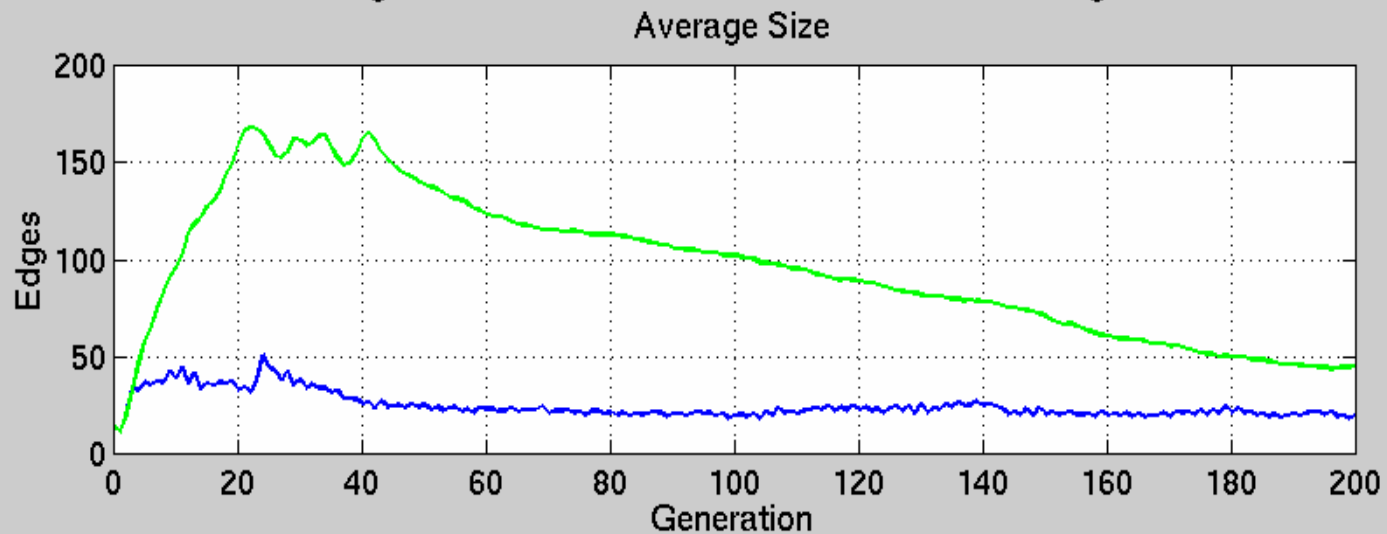
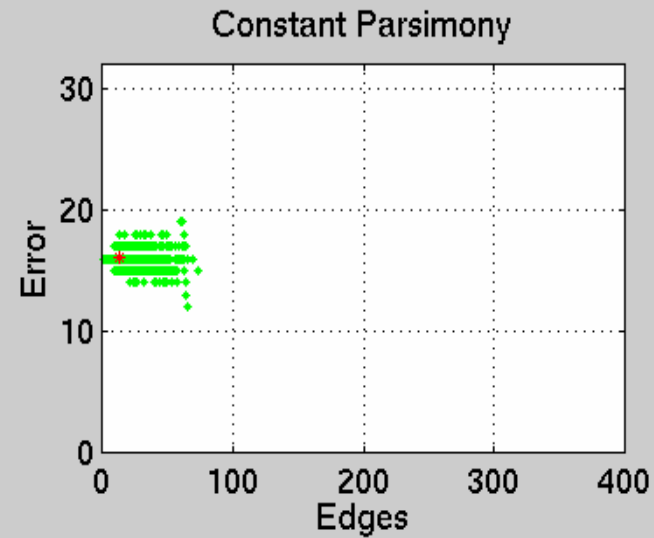
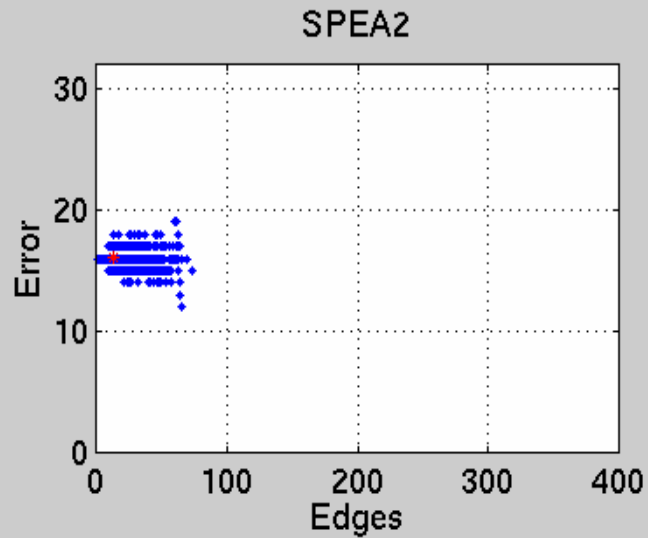
- large reduction of bloat
- smaller solutions
- solutions often found in fewer evaluations

Overview

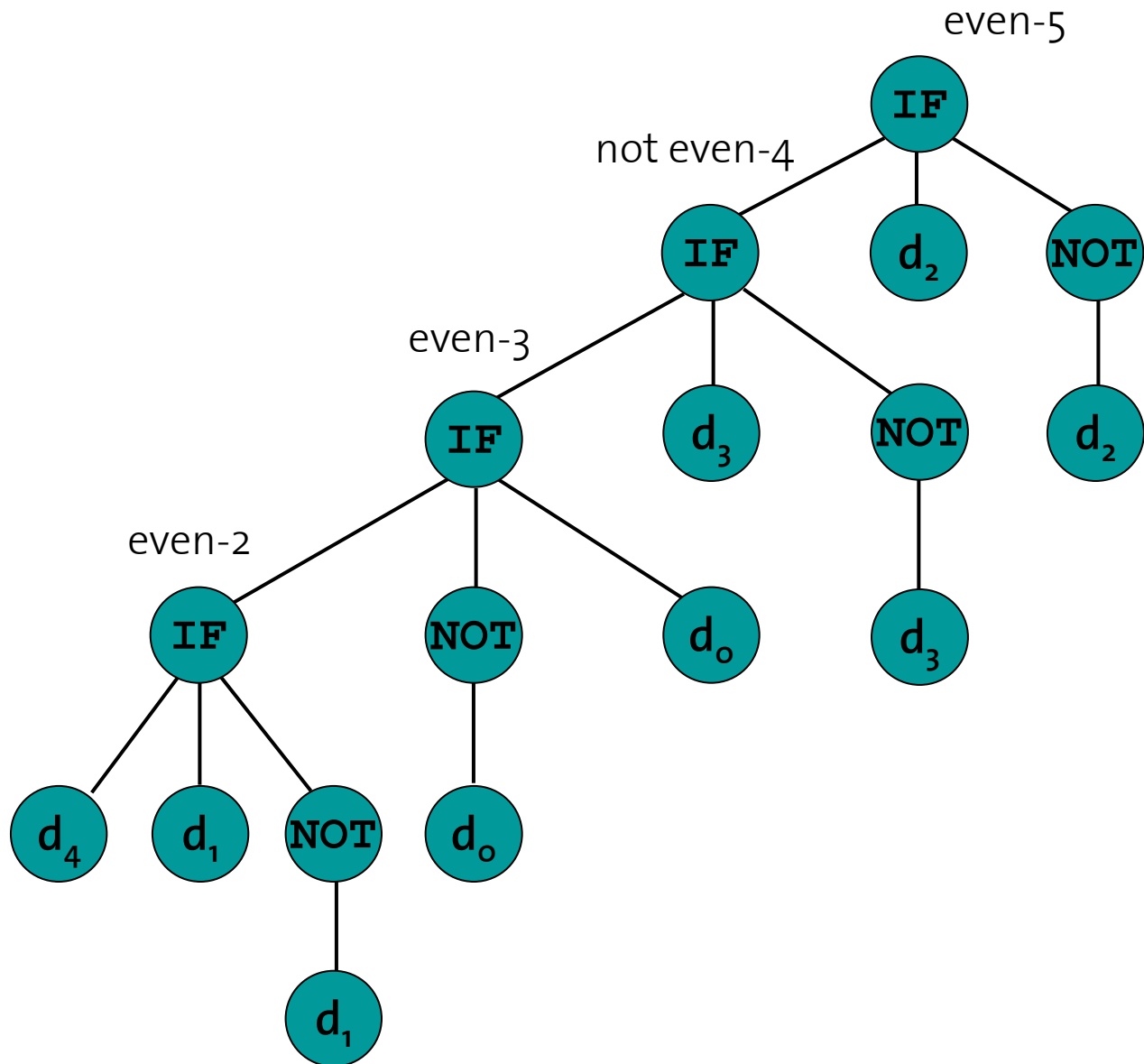


- 1 Traditional Methods
- 2 Multiobjective Approaches
- 3 Why are they effective?
A preliminary answer.**

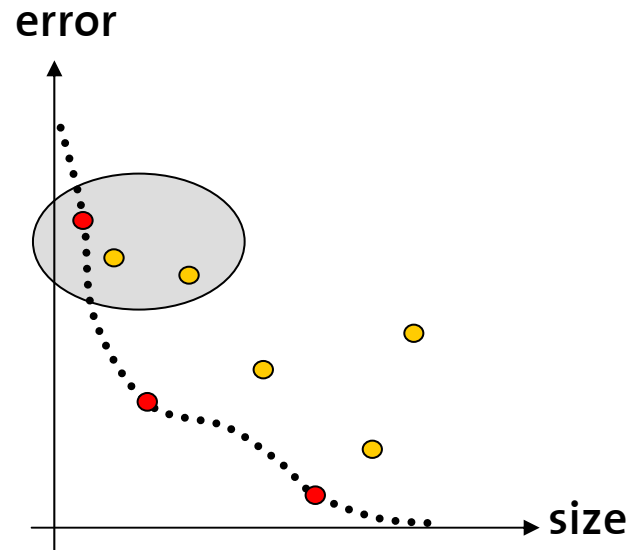
Results – Distribution of Populations



Ideal Trees



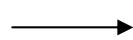
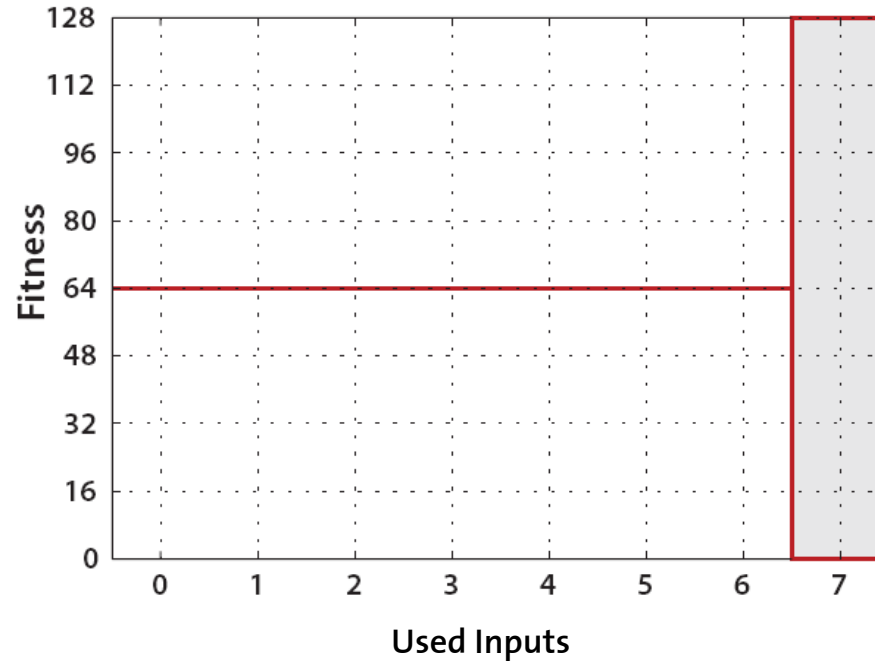
Building Block Hypothesis



Compact correct solutions could be composed of small, relatively fit individuals.

Testing the Building Block Hypothesis I

A closer look at the parity bit problem...

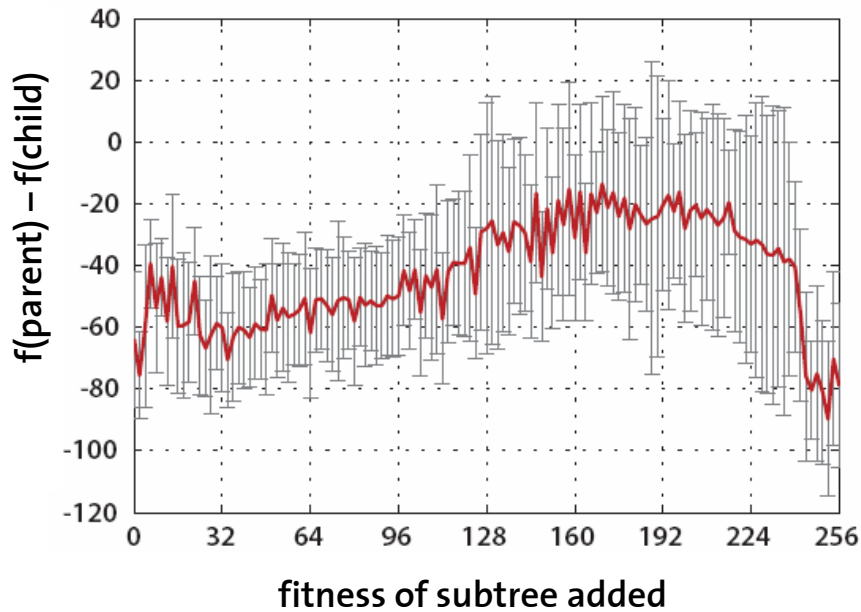


A correct k-parity solution is not good for $k < n$.

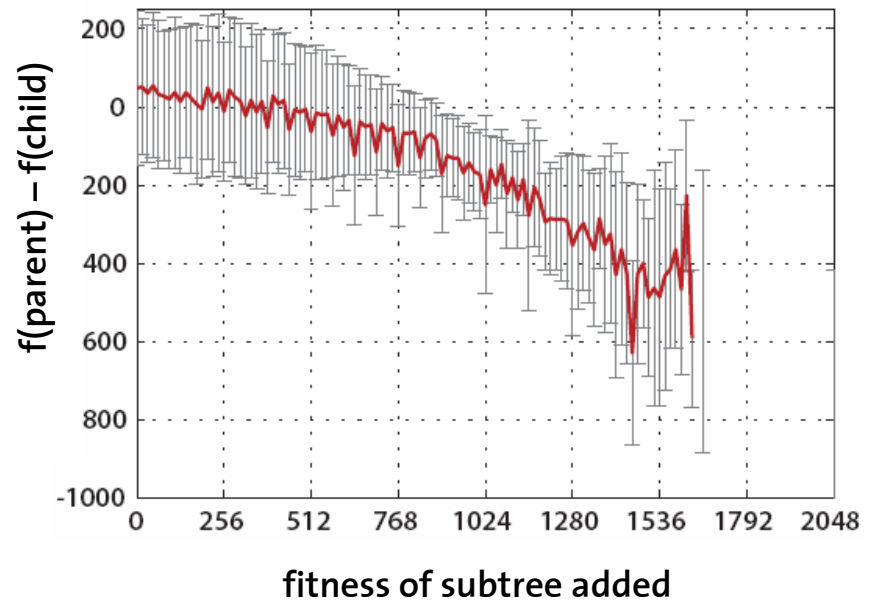
Testing the Building Block Hypothesis II

Are good solutions produced by adding solutions to subproblems?

8 bit hamming distance



11-multiplexer

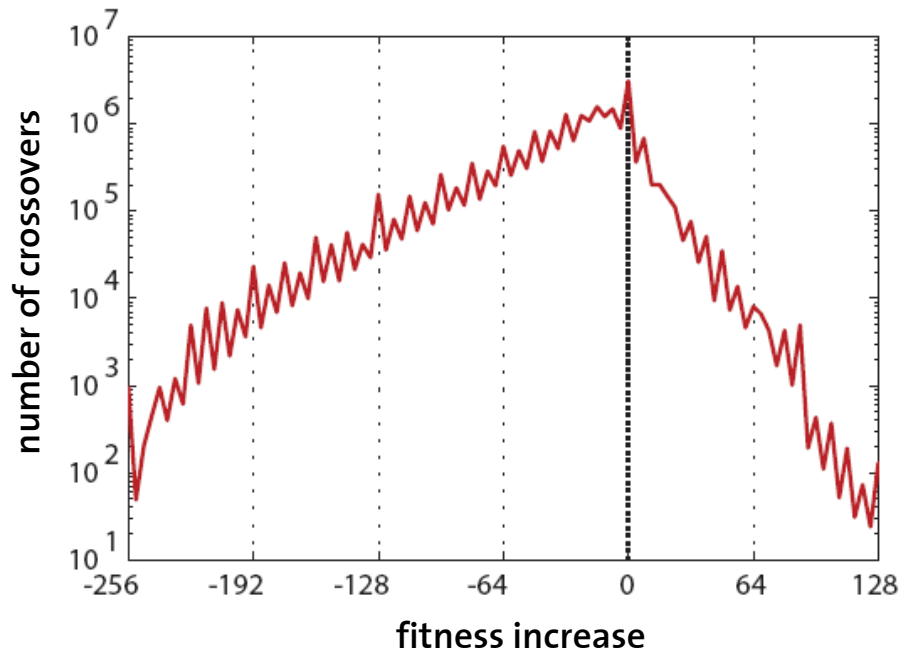


→ Increase in fitness does not happen by adding solutions which use only part of the inputs.

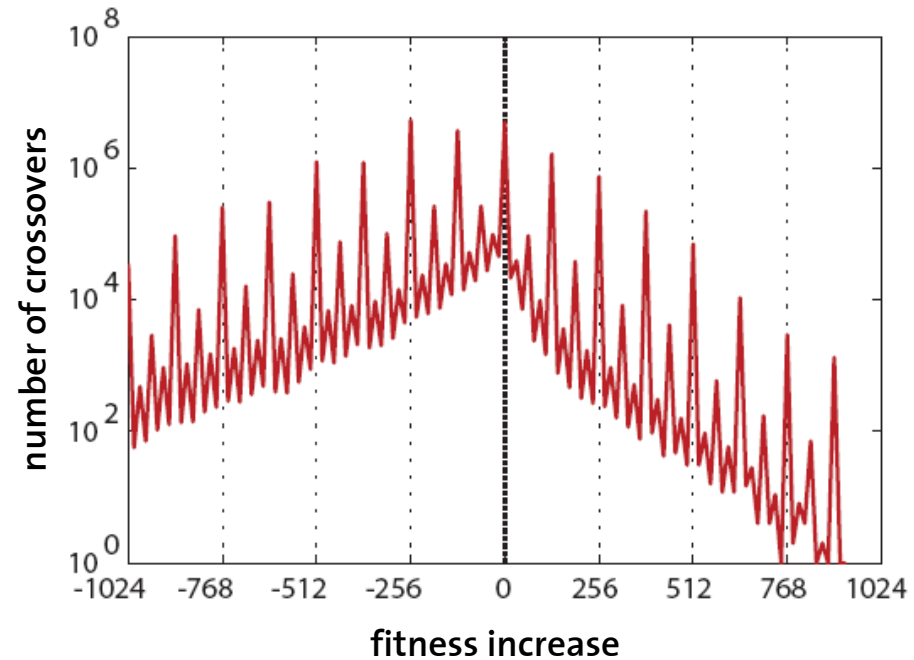
Testing the Building Block Hypothesis III

Does fitness often make jumps?

8 bit hamming distance



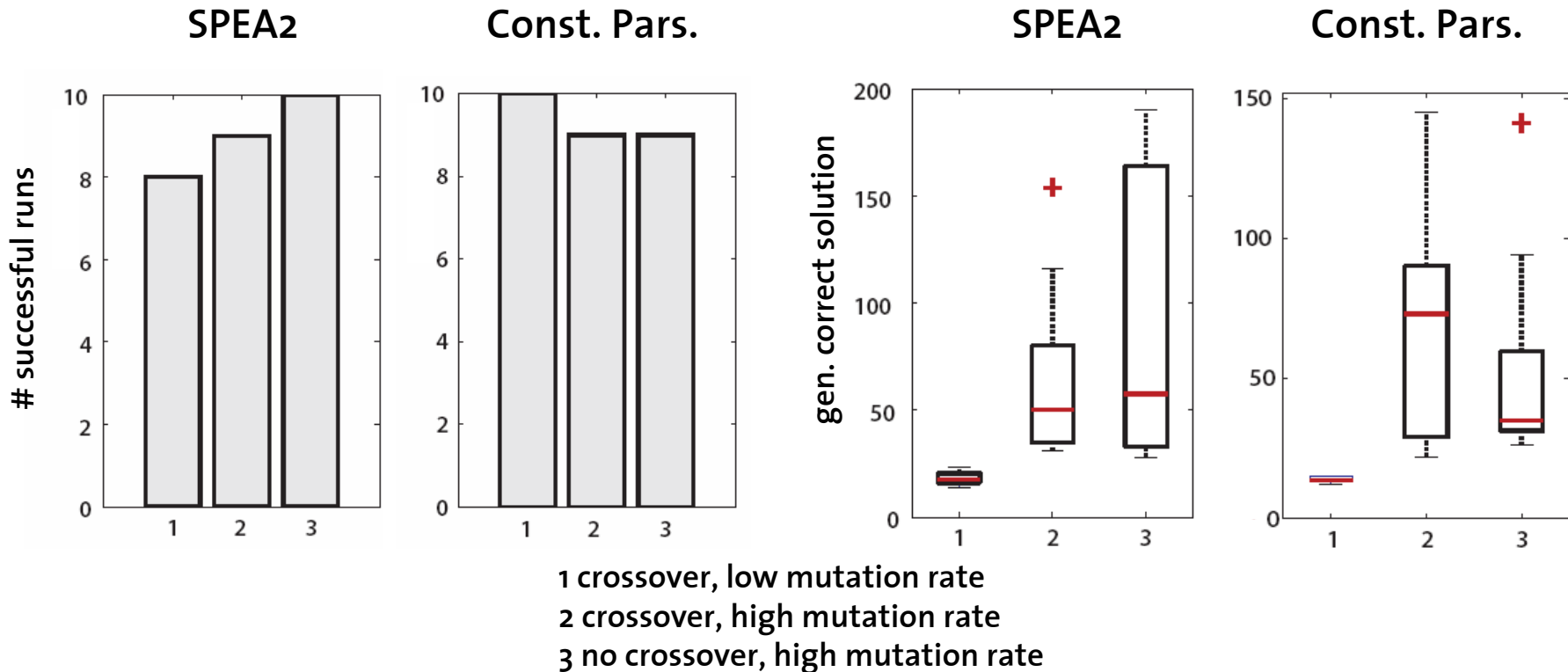
11-multiplexer



→ Big jumps in fitness are very rare.

Testing the Building Block Hypothesis IV

Switching off crossover...



→ SPEA2 does not loose much more than const. parsimony.

Conclusions

What is clear...

- pareto optimization is a valid approach to fight bloat
- average size of individual is reduced => lower CPU time
- size of correct solutions is reduced
- often solutions are found in fewer evaluations

What still needs some investigation...

- small individuals seem not to act as building blocks
- success seems to be based on maintaining diversity