CREST

Long-Term Evolution in Genetic Programming Computer Science, Aston University

2pm 7th March 2017

W. B. Langdon

Department of Computer Science

UCL

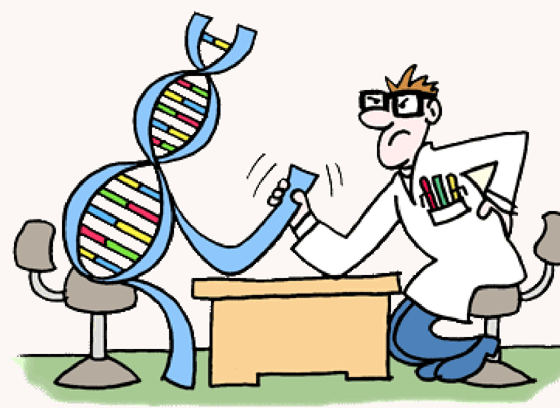
GI 2017, Berlin

deadline 29th March

GECCO workshop

Humies

$10000 Human-Competitive Results



Long-Term Evolution in Genetic Programming

W. B. Langdon

Computer Science. University College, London

GI 2017, Berlin

deadline 29th March

GECCO workshop

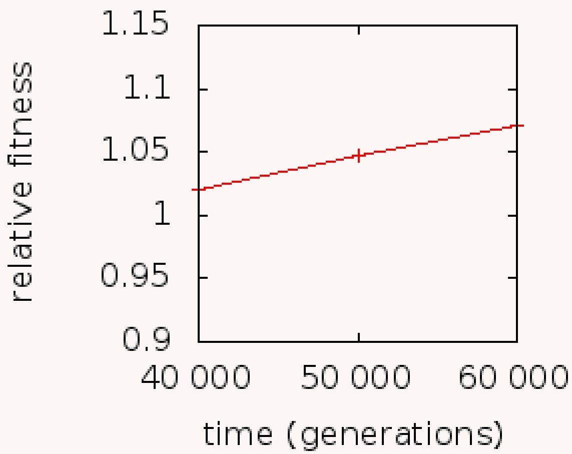
500 trees after
2000 generations

Genetic Programming and Long-Term Evolution Experiments

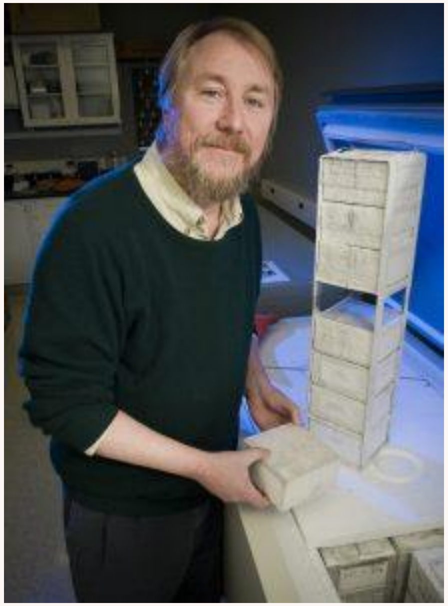
- Evolving Bacteria 60,000 generations v. evolving programs 100,000 generations
- LTEE continuous innovation v convergence
- Intro what is genetic programming:
  - GP is artificial evolution of functions
- Results
  - Increase in code (bloat), end of bloat
  - Theory some true, some less so

CREST

Long-Term Evolution Experiment



Mean fitness of nine E. coli populations from the LTEE



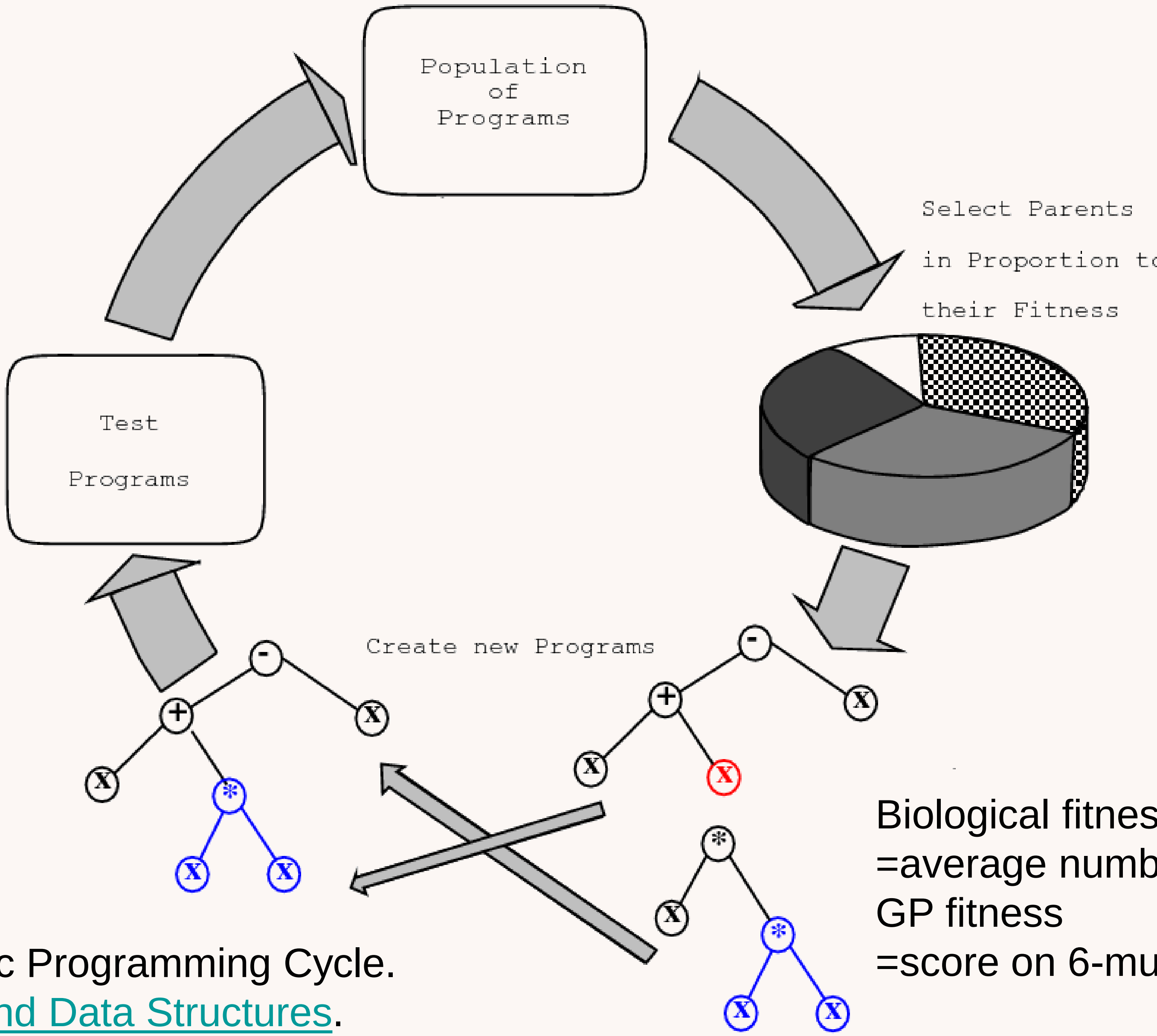
Richard Lenski pulls frozen bacteria cultures out of a freezer 15 Oct 2009

Evolving Bacteria 60,000 generations

Even after 60000 gens fitness still improving

R. E. Lenski *et al*. 2015. Sustained fitness gains and variability in fitness trajectories in the long-term evolution experiment with Escherichia coli. Proc. Royal Soc.

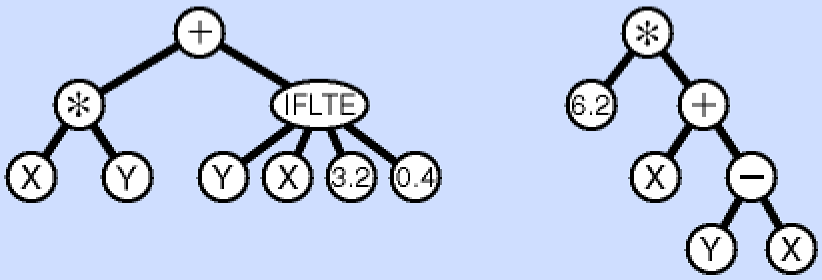
Artificial Evolution of Programs



Biological fitness
=average number of children
GP fitness
=score on 6-mux problem

The Genetic Programming Cycle.
From GP and Data Structures.

Creating new child programs: crossover



Crossover is symmetric.
That is, on average size after crossover = size before crossover

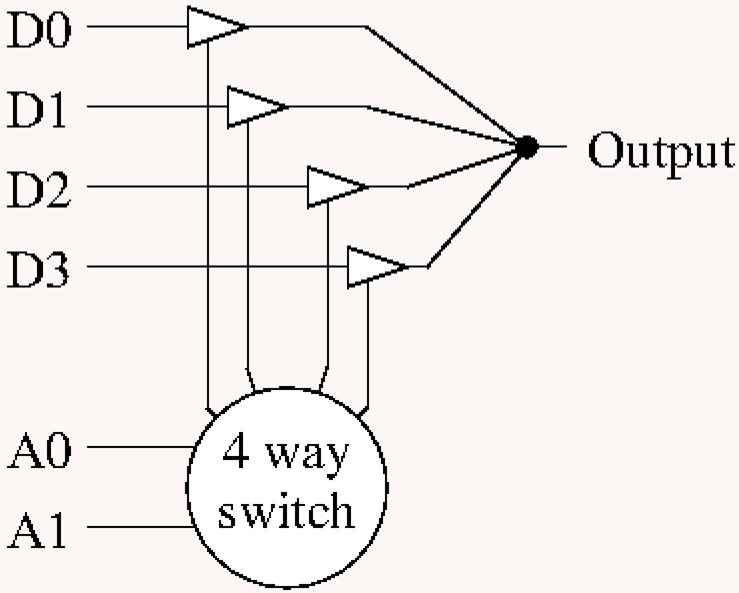
Genetic Programming and Long-Term Evolution Experiments

- GP system able to run thousands of generations. (Do not stop when solved)
  - Expect bloat (tree growth)
  - Compact representation of trees
  - Fast fitness evaluation
- Submachine code genetic programming

GPquick

- GPquick C++, written by Andy Singleton
  ≈ two bytes per tree node
- Submachine code GP
  – Boolean (bit) problems.
  – AND, NAND, OR, NOR operate simultaneously in parallel on bits in word (e.g. 32 or 64 bits)
  – 64 bit computer can do 64 test cases in parallel

6 Multiplexor



- GP bench mark.
- Six inputs:
  - Use two (D4 D5) as binary number to connect corresponding data lines (D0-D3) to the output
- Test on all 2^6=64 possible combinations
- Fitness score (0-64) is number correct

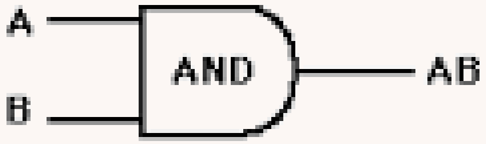
Genetic Programming to solve 6-Mux

- Terminals (tree leafs)
  - D0,D1,D2,D3 D4,D5
- Function set: 2 input gates→ binary trees
  - AND, NAND, OR, NOR. No side effects
- Generational population of 500 trees
- Tournament selection: choose best of 7
- 100% subtree crossover
- Initially hard limit on tree size (10^6)

Impact of Subtrees

- Subtree like whole tree.
- Output of subtree is via its root node
- **Intron**: subtree which has no effect on overall fitness. I.e. its output does not impact on root node of whole tree.
- **Constant** subtree always has same output, i.e. same output on all 64 test cases.
- Remaining **effective code** has an impact on root node. Typically it is next root node

Example Intron: AND Function



A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

A	B	Out
0	0	0
~~0~~	~~1~~	~~0~~
1	0	0
~~1~~	~~1~~	~~1~~

Left: two input AND node.

Right: same but input B is always 0.

So output always 0. Input A has no effect.

Subtree A is always ignored, even in child.

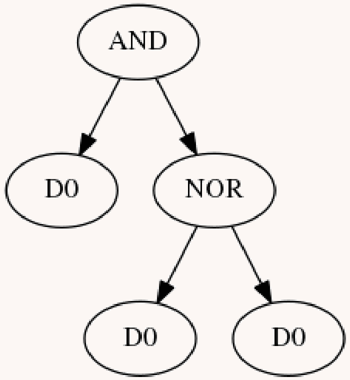
(NB no side effects)

CREST

Constants

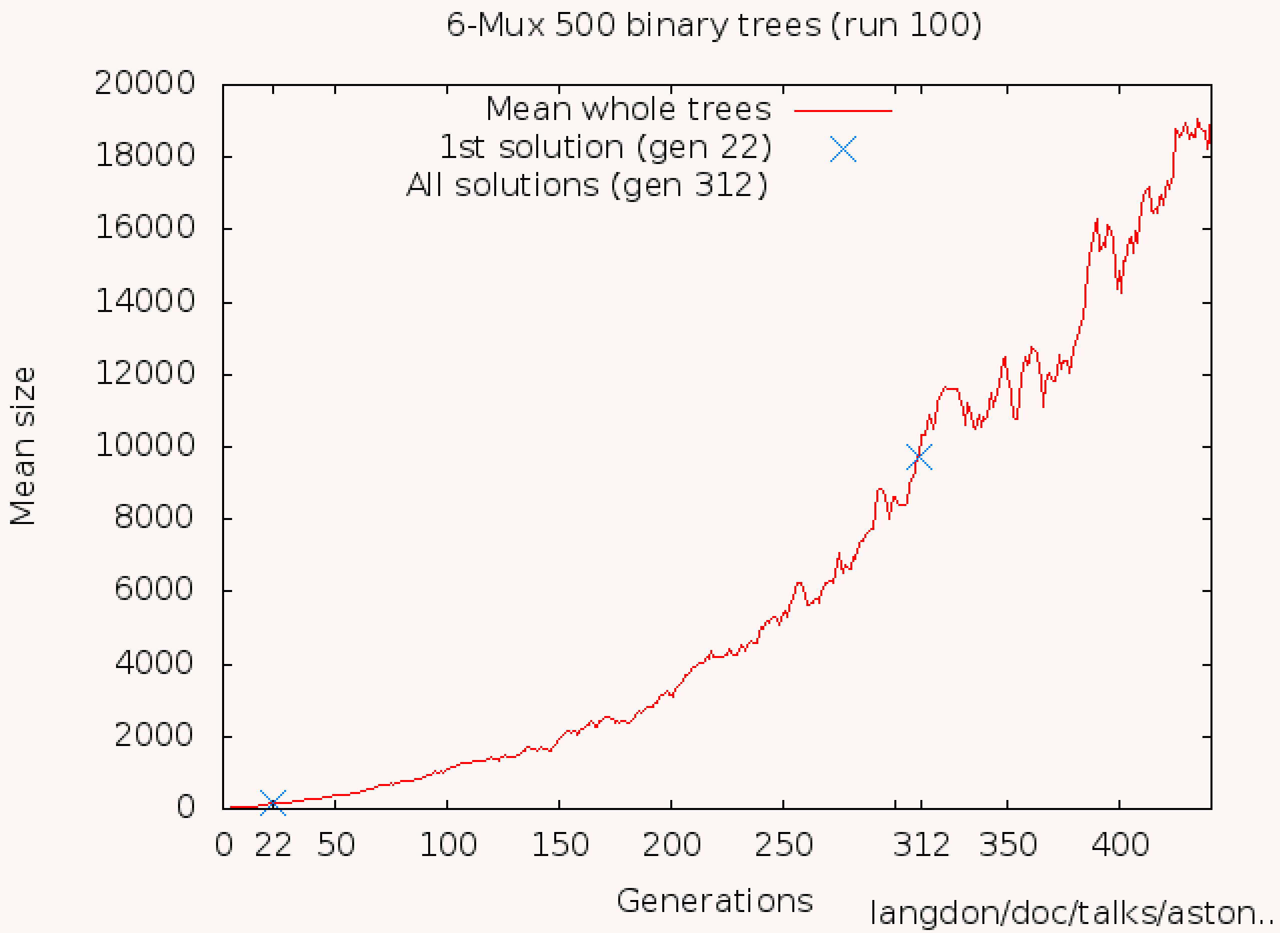
- Two constants: always 0 and always 1 (FFFFFFFFFFFFFFFF).
- E.g. evolve by negating input and ANDing with same input

  (AND D0 (NOR D0 D0)) = 0



- Constants help form introns but may be disrupted by crossover.
- However large subtrees which always output either 0 or 1 tend to be resilient to crossover

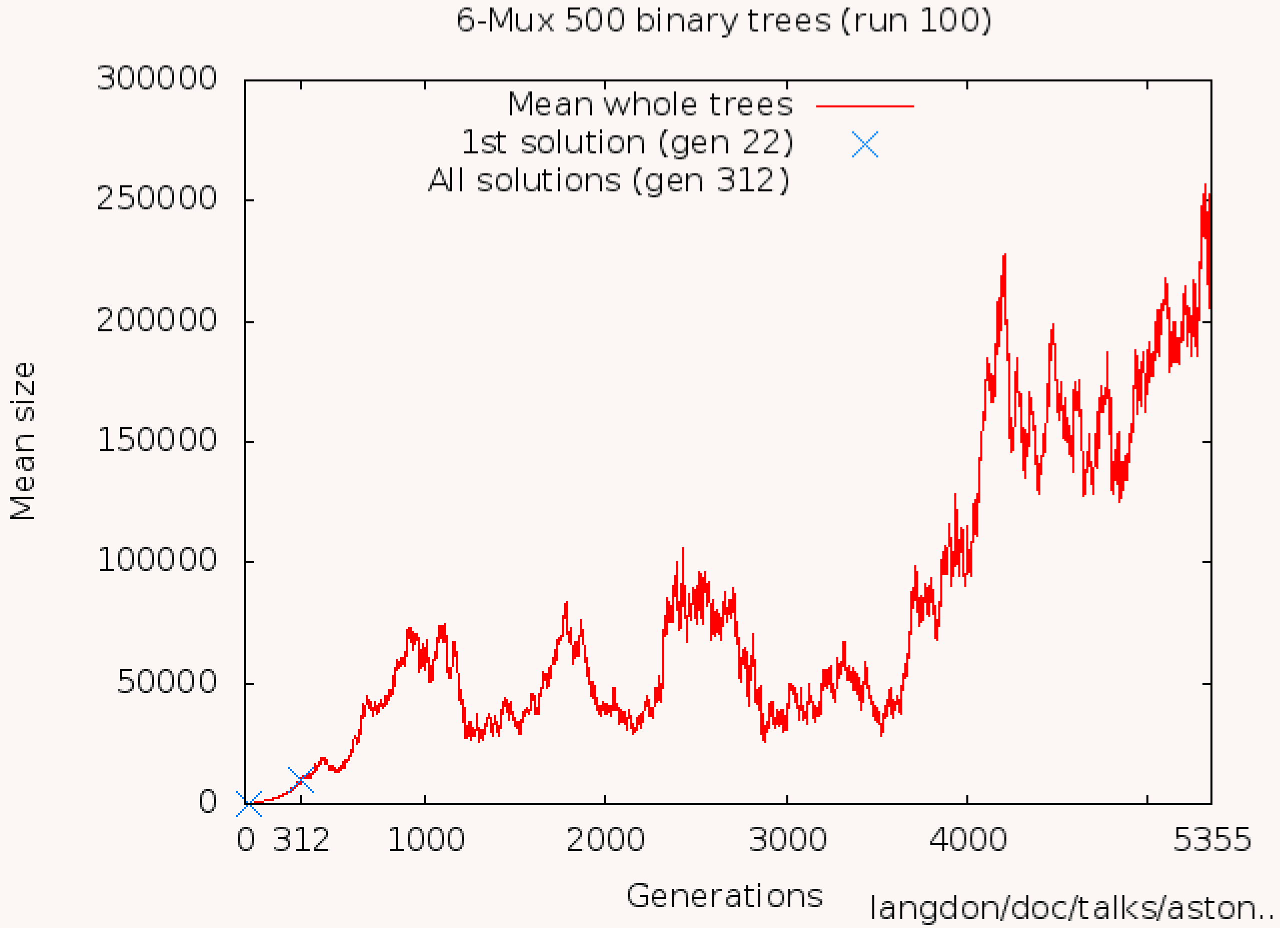
LTEE evolution of size



Note evolution continues after 1st solution found in generation 22 and even after 1st population where everyone has maximum fitness (generation 312).

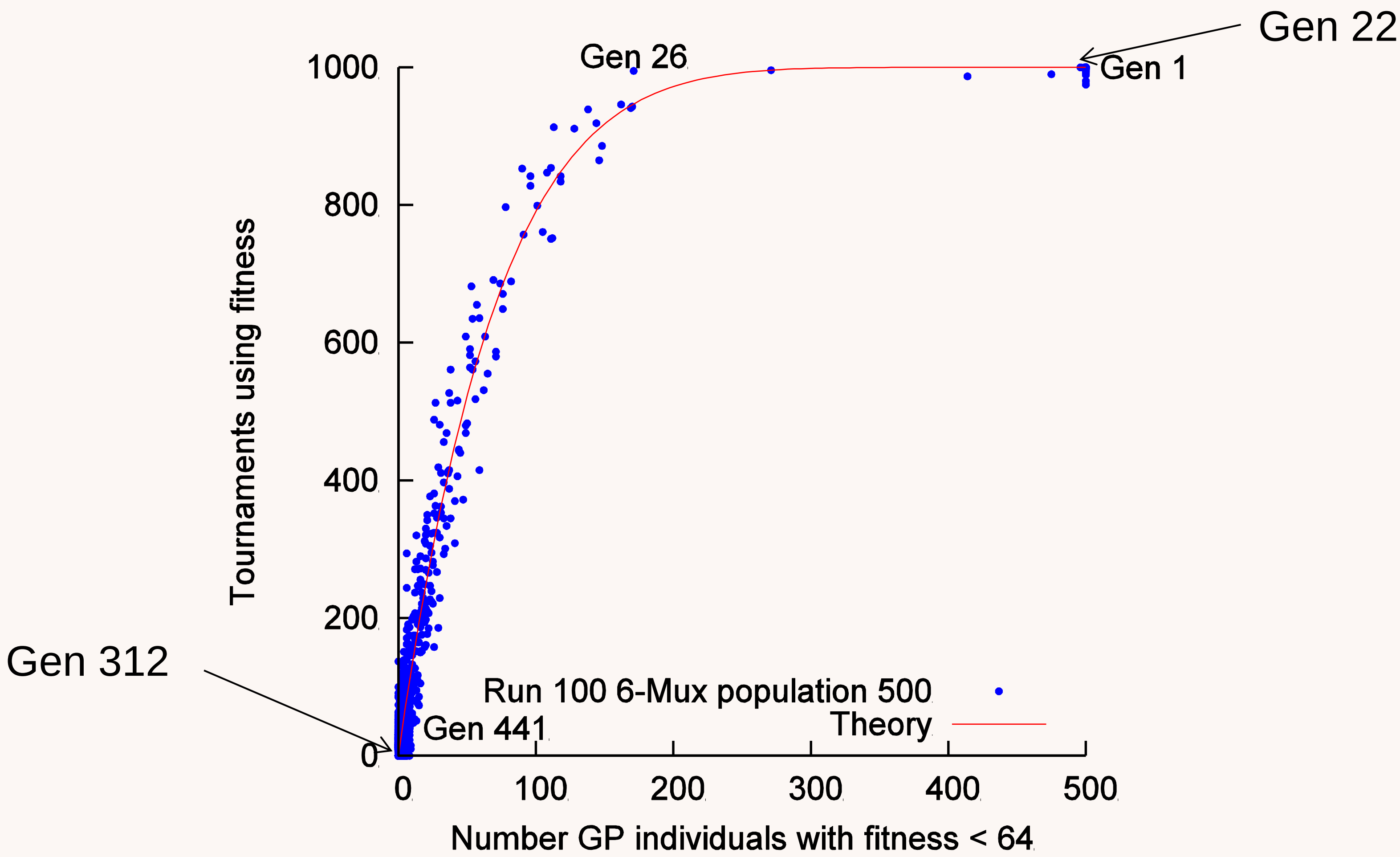
CREST

LTEE evolution of size



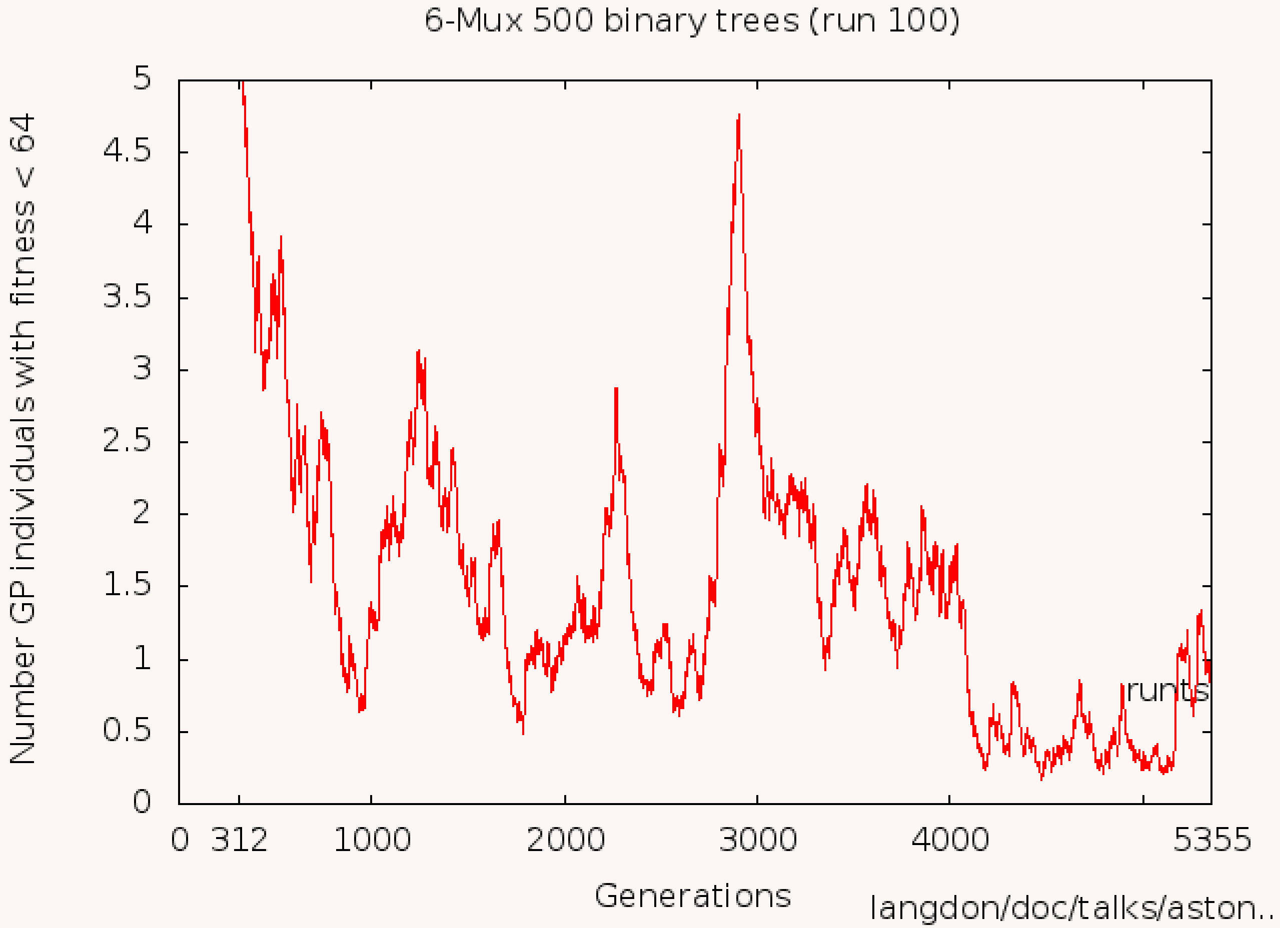
Note evolution continues even after 1st population where everyone has maximum fitness (generation 312) but falls as well as rises.

6-Mux Fitness Convergence



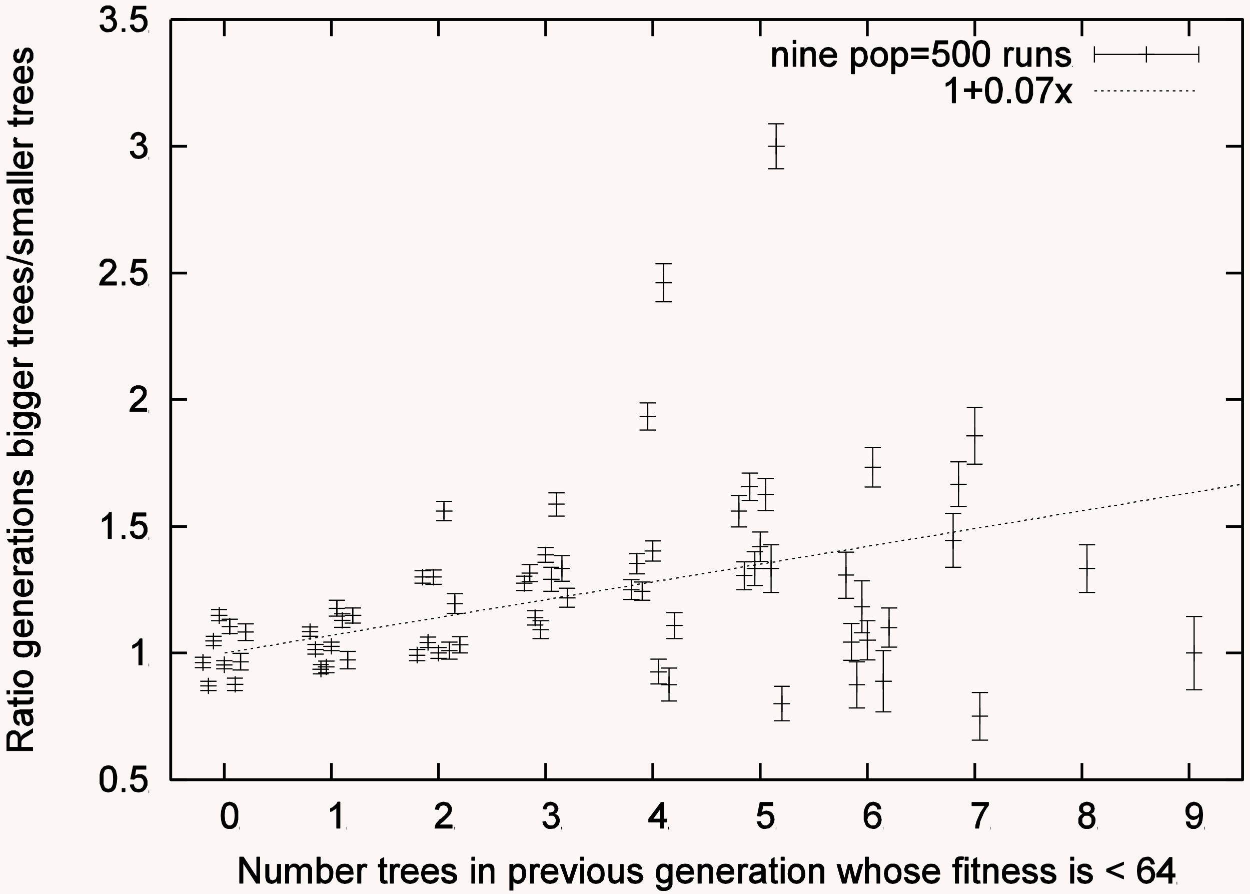
Theory $y = 2\text{popsize}(1-(1-x/\text{popsize})^7)$ matches experiment

6-Mux Fitness Convergence



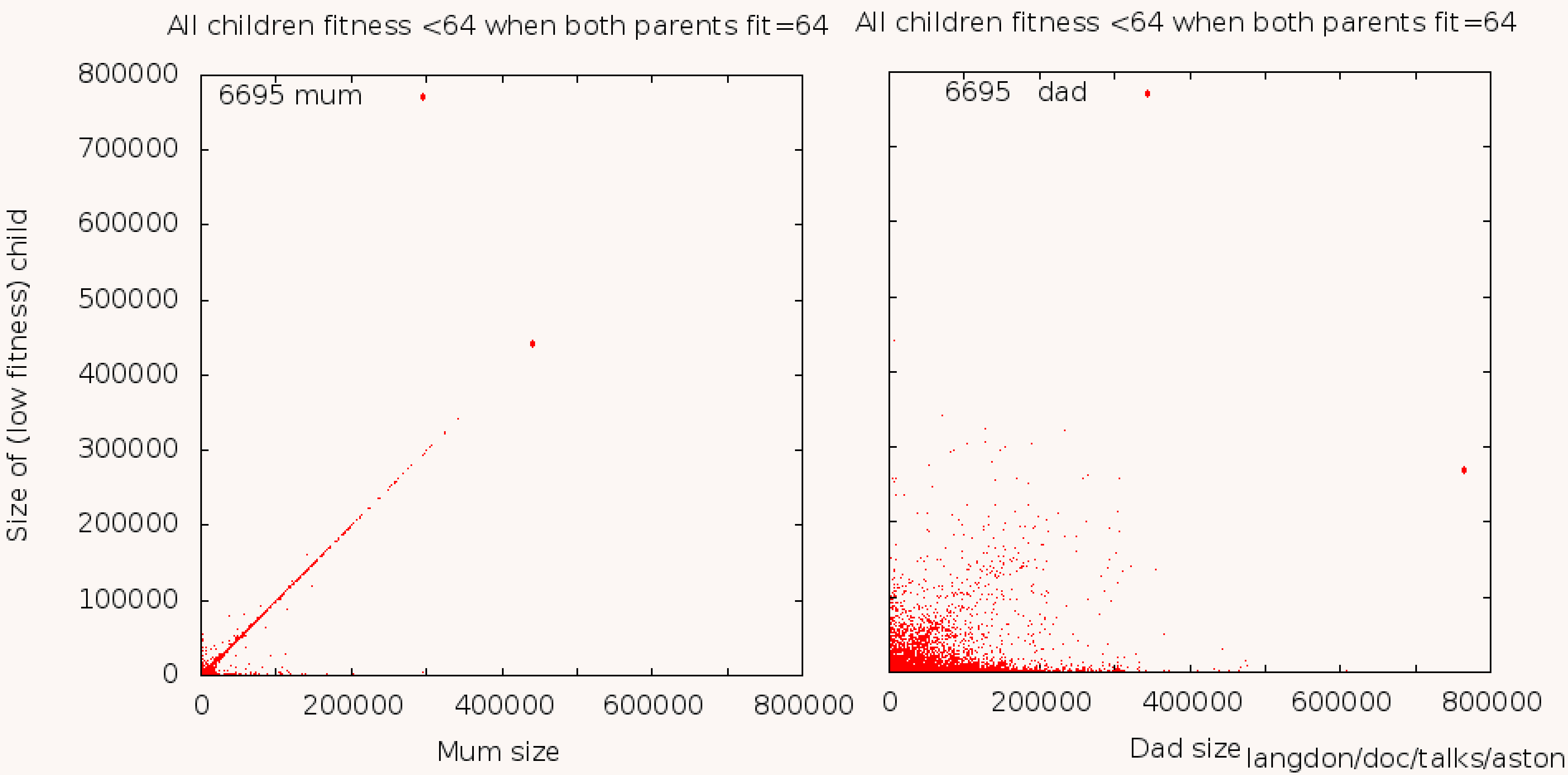
Plot smoothed by taking running average over 30 generations

Runts Drive Evolution



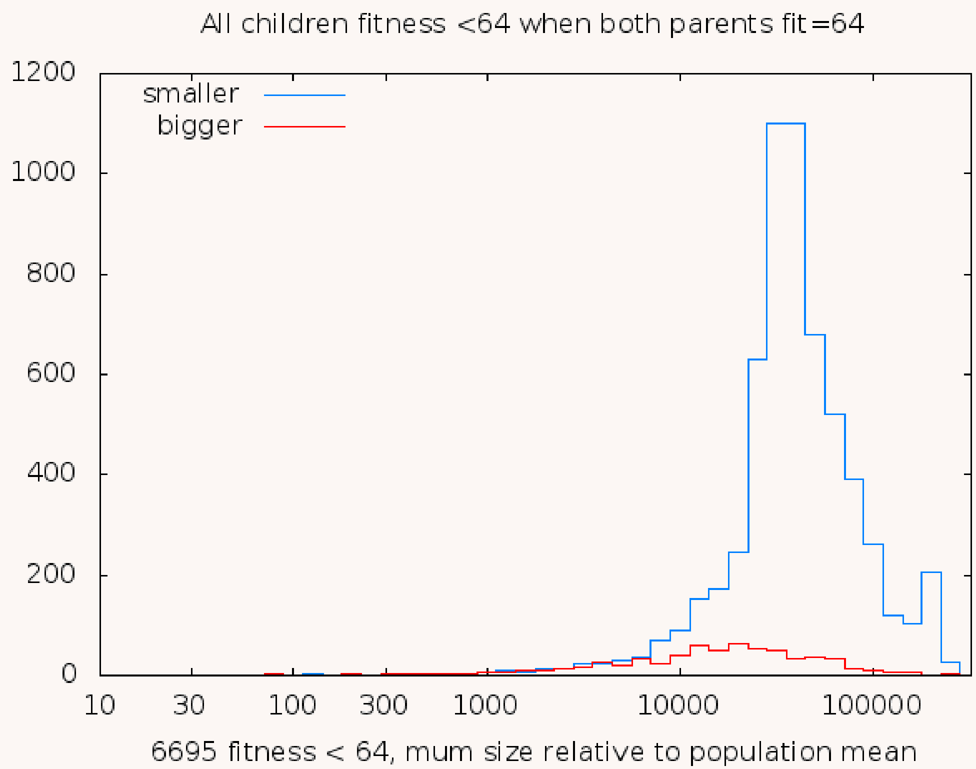
Don't plot ratio if less than 5 data

Importance of Mothers



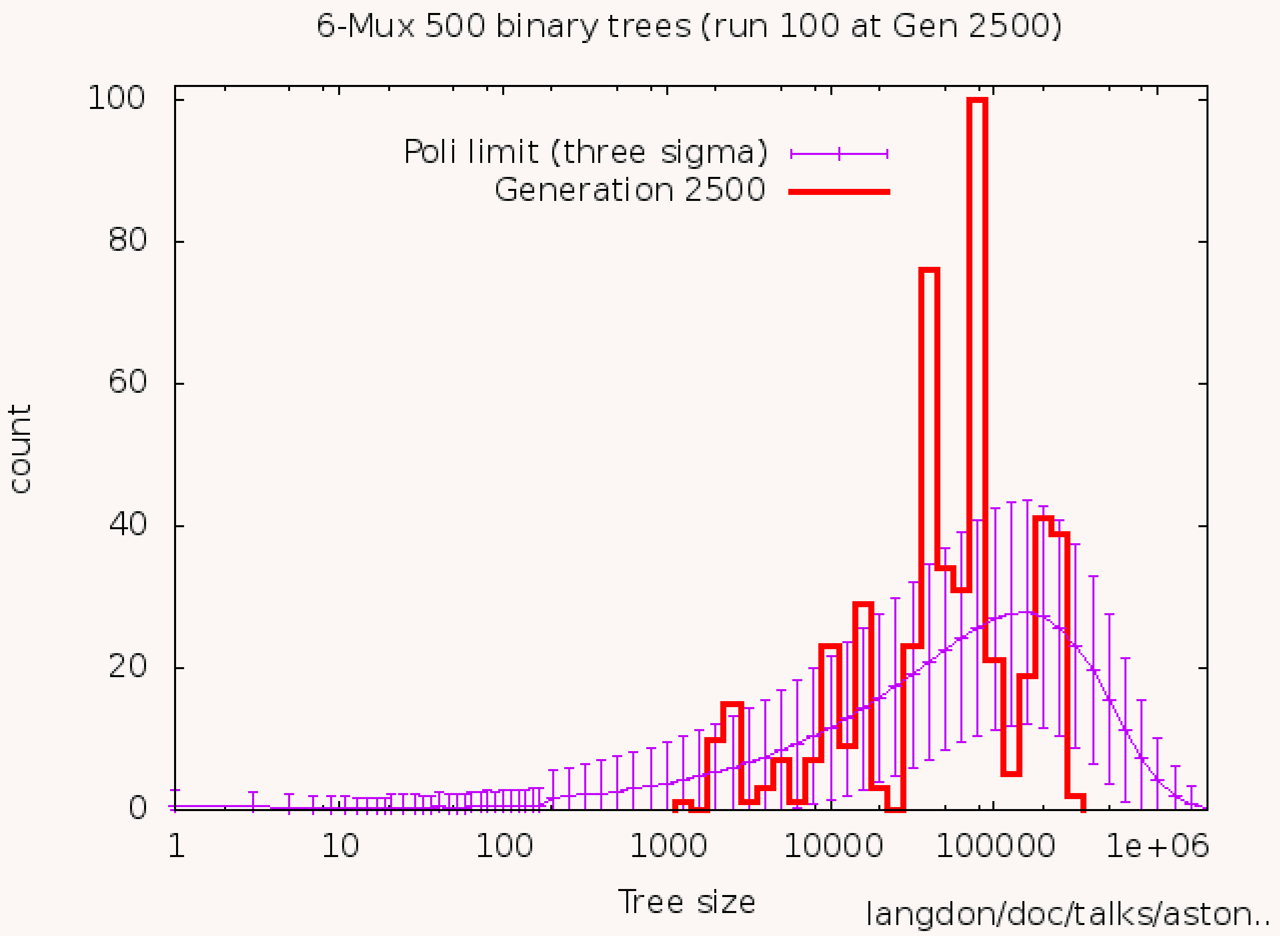
Size of poor fitness children closely related to parent who they inherit root from (mum).

A few runts drive size increase



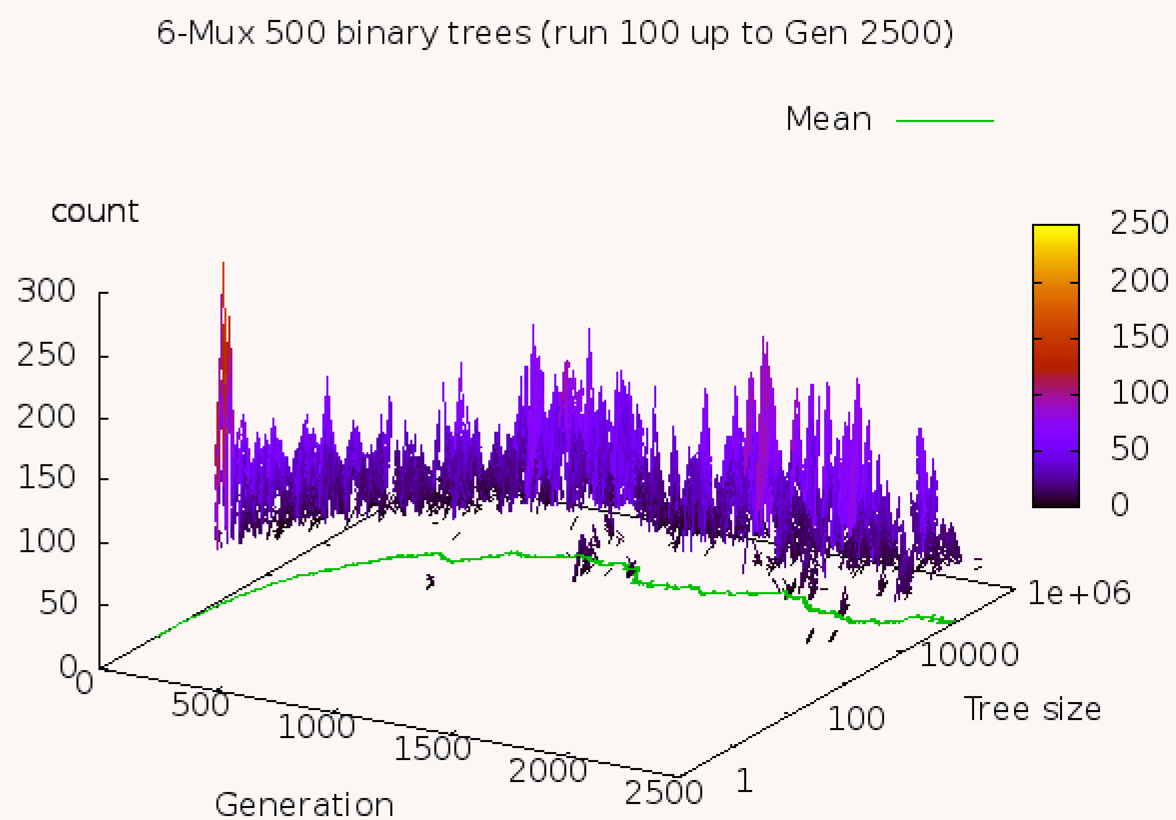
- Many mothers of runts are smaller than average (blue)
- Selection removes all low fitness children (runts)
- Since these are smaller than average
- Although there is noise, on average size increases

Testing Theory



- Theory assumes crossover only (no selection). In earlier work distribution of sizes converged to limit rapidly.
- Selection caused by a few runts modifies size distribution

Testing Theory



- Same as testing theory plot but do every generation
- Colour only part of histogram ≥ 3σ
- Small tree and large tree tails ok (not coloured)

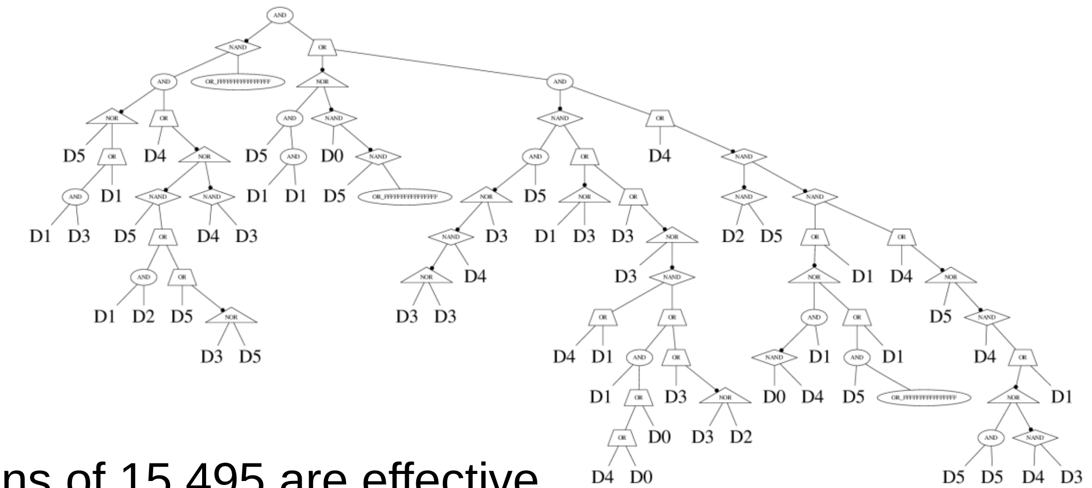
Convergence in Genetic Programming

- GP genotypes typically do not converge. Even after many generations every tree in the population is different, BUT…
- Every (or almost all) trees give the same answers (phenotypic convergence)
- Effective code, i.e. code to solve problem, does converge.

Effective code other runs converges differently

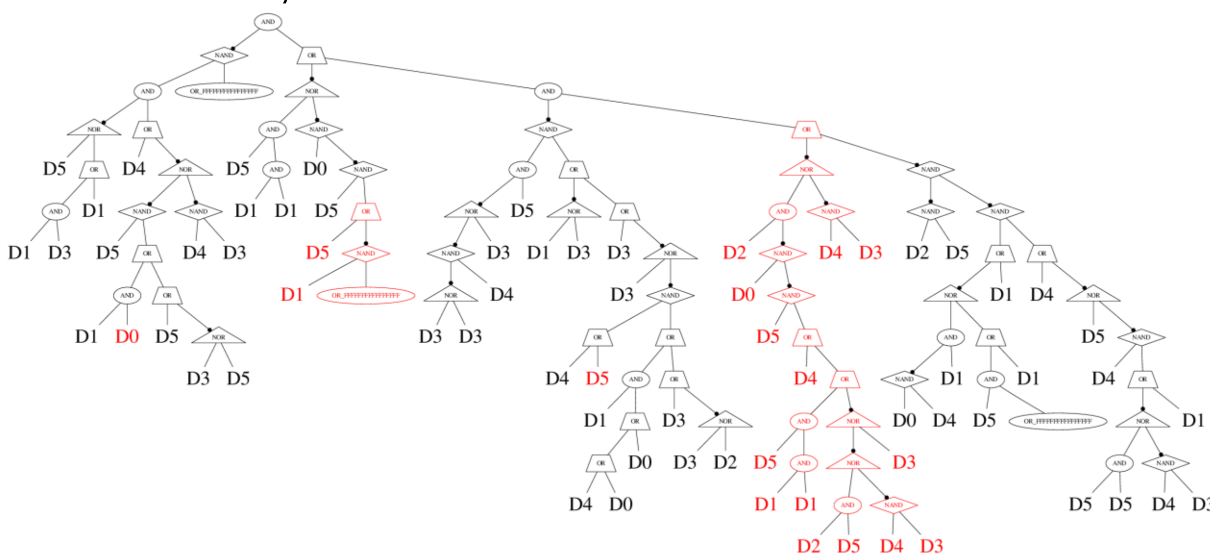
CREST

Convergence of typical Effective Code



Gen 400

Only 111 instructions of 15,495 are effective

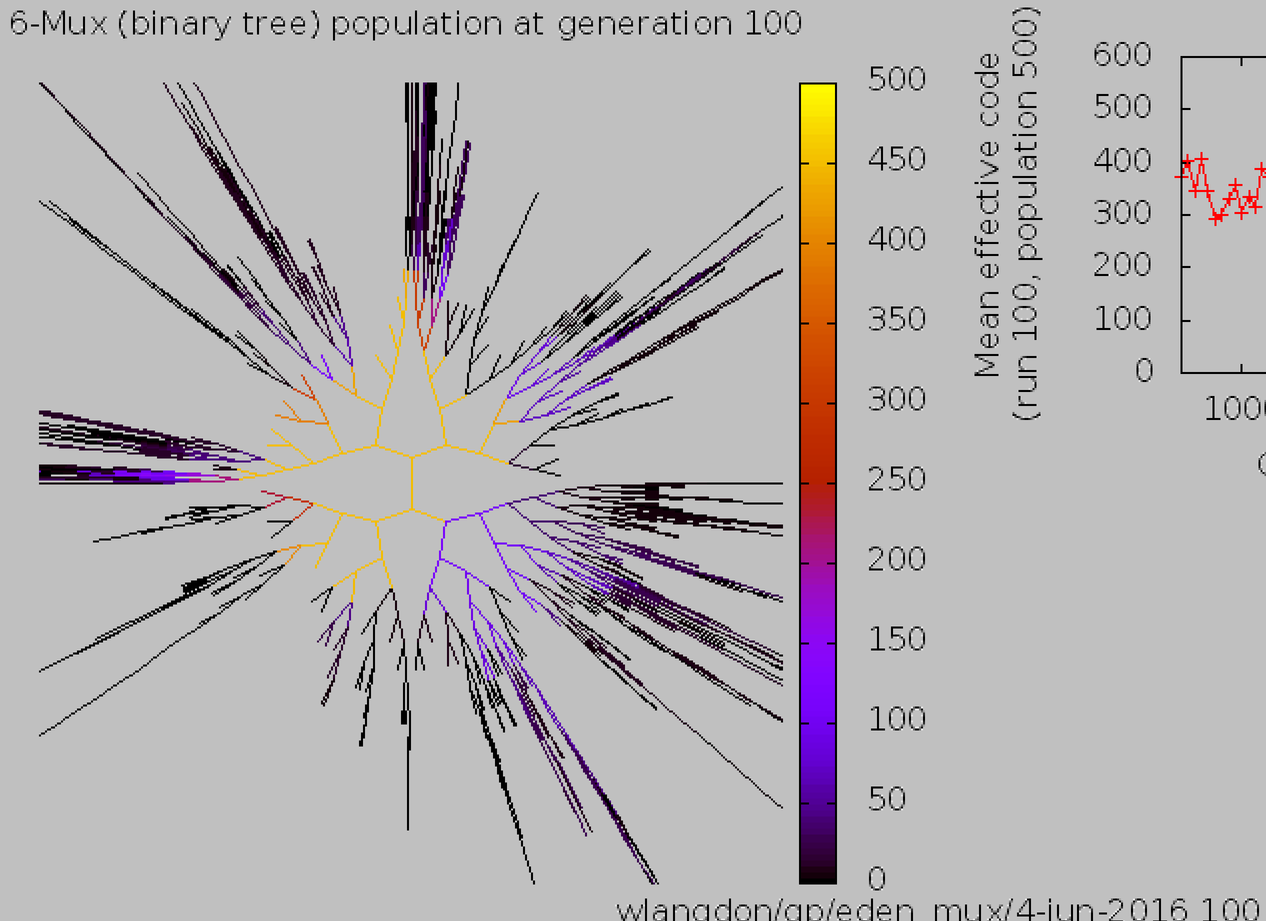


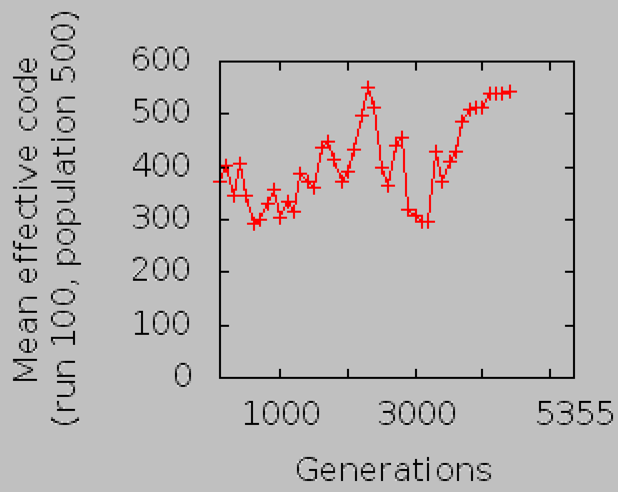
Gen 500

Only 141 instructions of 16,831 are effective

Tree drawing code lisp2dot.awk

Convergence of Effective Code



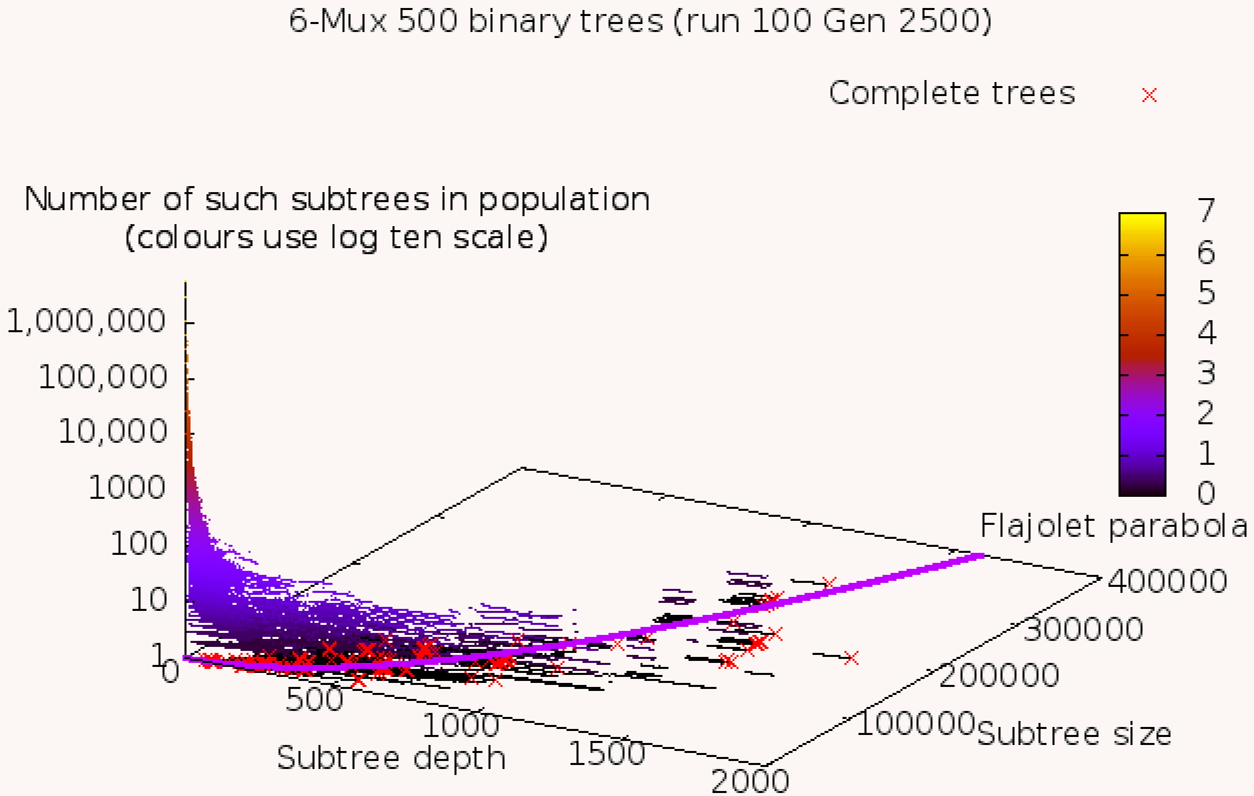


Effective code only. Yellow highly converged.

Black unique code

Circular lattice code [gp2lattice.awk]()

Shapes of Evolved Trees



Both whole trees × and subtrees lie near Flajolet Depth $\approx 2\left(\frac{\pi\, size}{2}\right)^{1/2}$ limit[1] for random trees

CREST

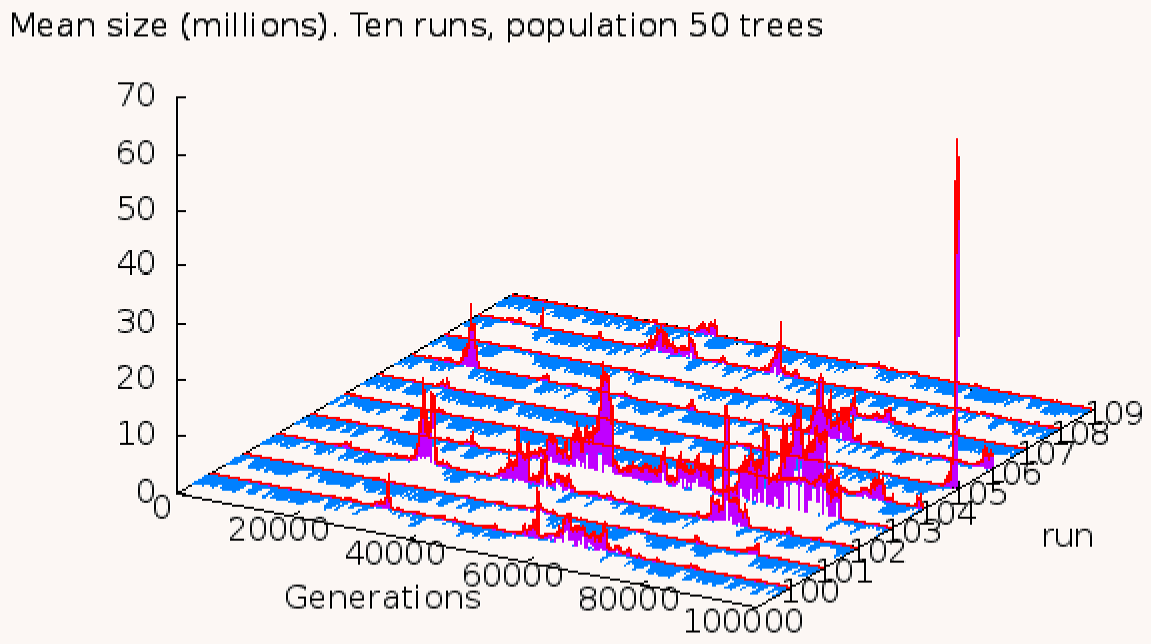
Bloat limited by Gambler's ruin

- Tiny fraction of disrupted (low fitness) children sufficient to drive evolution towards every bigger trees.
- As trees get bigger chance of hitting protected effective code near root node falls.
- In a finite population eventually no child will be disrupted.
- Size, without fitness, difference just wanders at random.
- Crossover cannot escape from population of tiny trees.
- So we have a lower limit on the random fluctuation.

  I.e. a Gambler's ruin.
- But wondering towards lower limit will re-establish the conditions for bloat.
- Very approximate limit on tree size:

  tree size ≈ number of trees × core code size

Bloat limited by Gambler's ruin



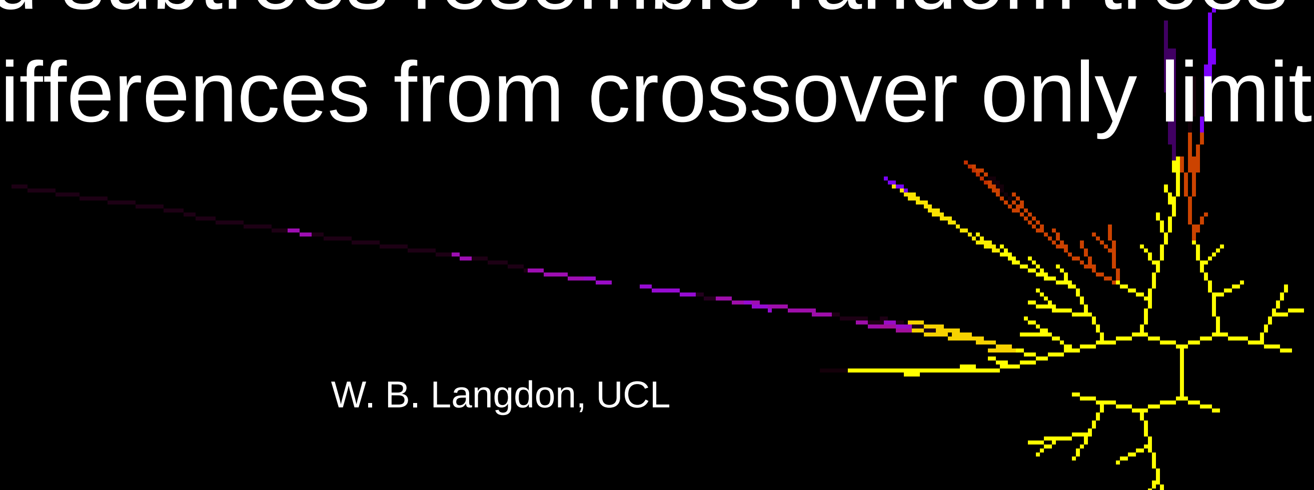
- tree size ≈ number of trees × core code size
- tree size ≈ 50 × 497 ≈ 25000
- Across ten runs and 100,000 generation, median mean size 42,507 (smallest tree in pop size=10,513)

In all ten runs the whole population repeatedly collapses towards smaller trees

CREST

Conclusions

- Studied long term evolution (>>any other GP)
- 100s gens where everyone has same fitness
- No selection to drive size increase
- Gambler's ruin with size falling as well as rising
- Evolved effective code surrounded by ring of sacrificial constants and introns
- Trees and subtrees resemble random trees
- But still differences from crossover only limit



END

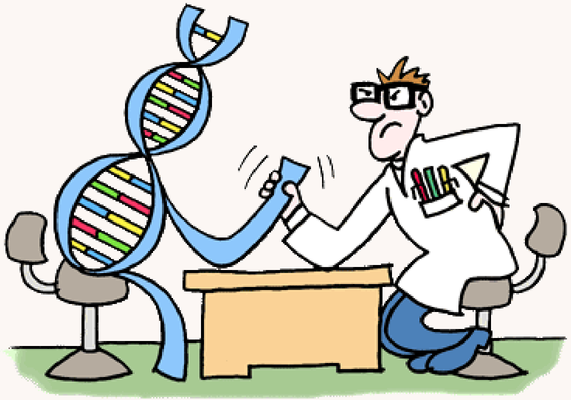
http://www.cs.ucl.ac.uk/staff/W.Langdon/ http://www.epsrc.ac.uk/ EPSRC

GECCO-2017 15th-19th July 2017 in Berlin

- Genetic Improvement workshop at Submissions by **29th March**
  http://geneticimprovementofsoftware.com/
- Total $10 000 prizes for human competitive results. Entries by 7th June
  http://www.human-competitive.org/call-for-entries

GI 2017, Berlin,
15/16 July 2017
GECCO workshop

Humies
$10000 Human-Competitive
Results

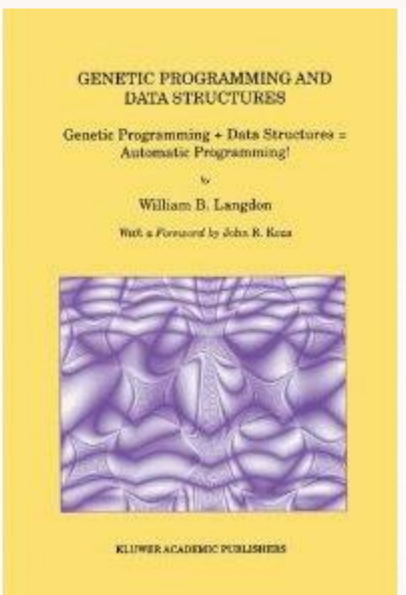


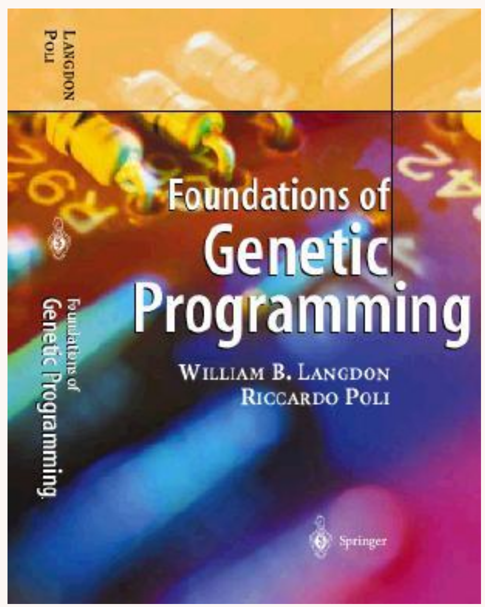
Genetic Programming

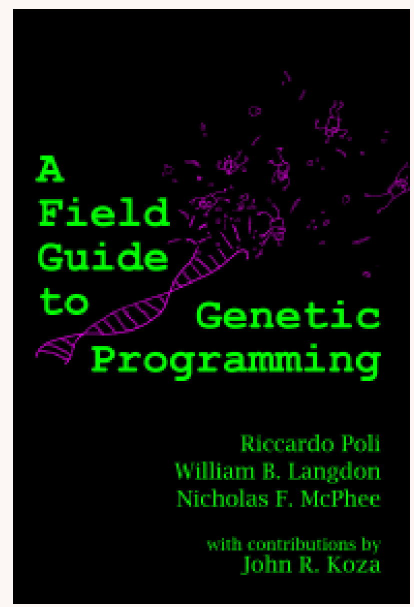
W. B. Langdon

CREST

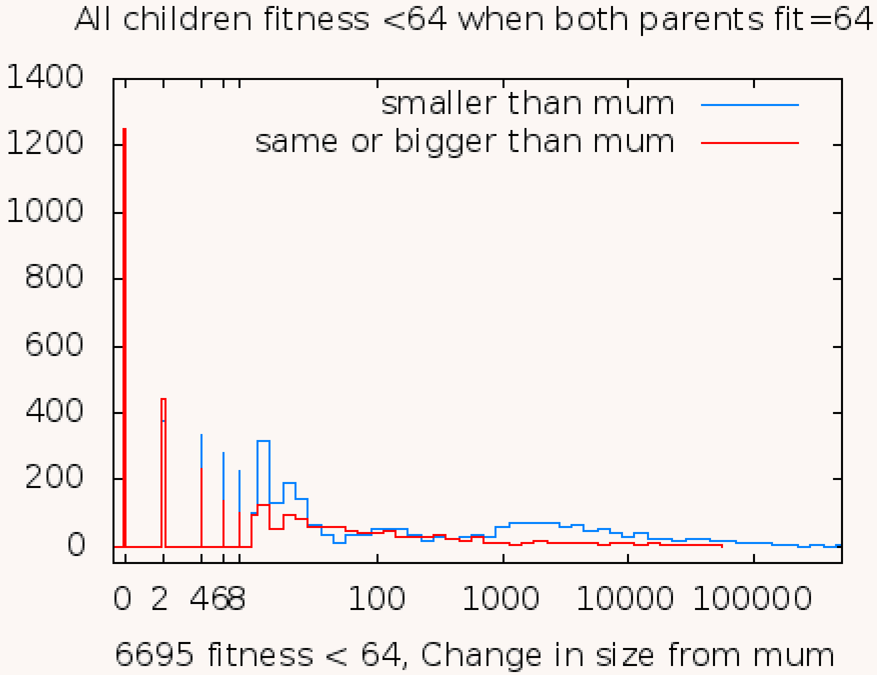
Department of Computer Science

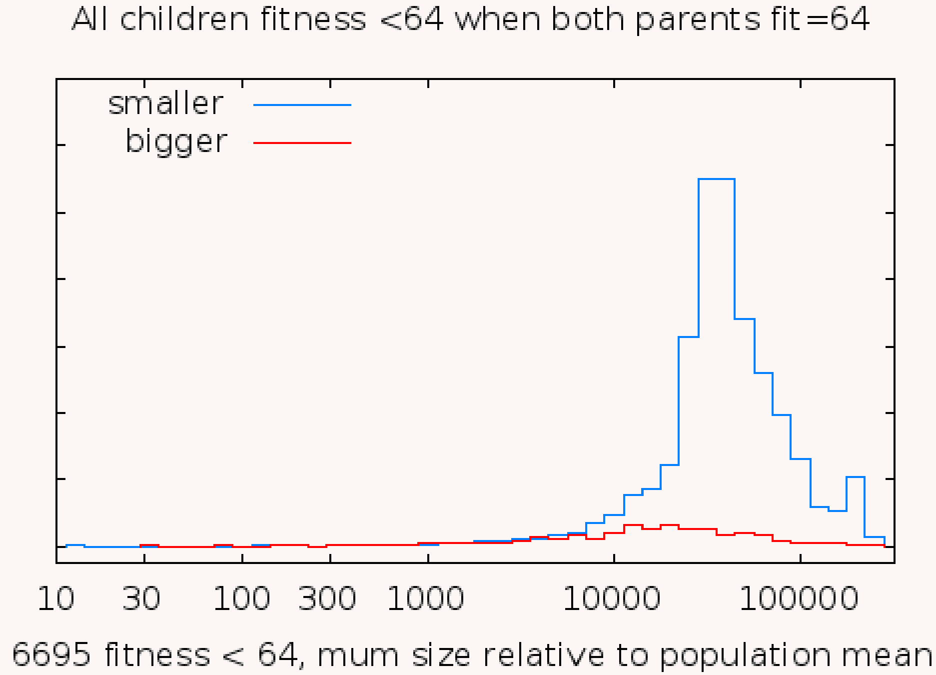






Importance of Mothers





- Although many runts are smaller than their mum,
- many mothers of runs are smaller than average.
- Selection removes all low fitness children,
- Since these are smaller than average, the average size increases

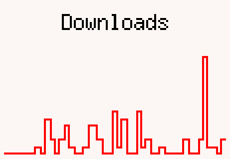
The Genetic Programming Bibliography

http://www.cs.bham.ac.uk/~wbl/biblio/

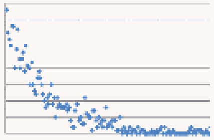
11322 references

RSS Support available through the
Collection of CS Bibliographies. XML RSS

A web form for adding your entries.
Co-authorship community. Downloads



A personalised list of every author's
GP publications.



bibliography

blog

Search the GP Bibliography at
http://liinwww.ira.uka.de/bibliography/Ai/genetic.programming.html