14th TAROT Summer School 2018
UCL, 2-6th July 2018

Genetic Improvement

W. B. Langdon
Computer Science, University College, London

WIKIPEDIA
Genetic Improvement

GI 18

Annual workshops on GI

A Comprehensive Survey, IEEE TEVC

Genetic Improvement

W. B. Langdon

Computer Science, University College, London

WIKIPEDIA

Genetic Improvement

GI 18

Annual workshops on GI

A Comprehensive Survey, IEEE TEVC

Genetic Improvement of Software

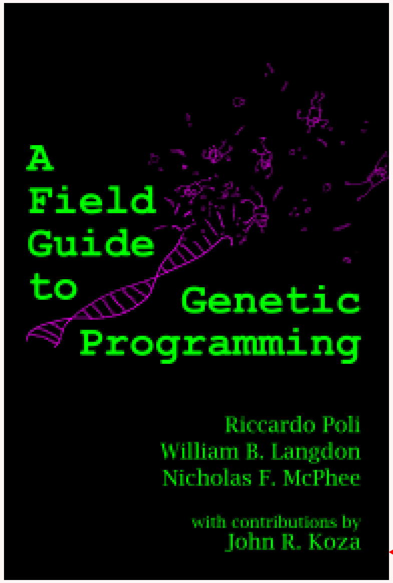
- What is Genetic Improvement
  - – Genetic Programming (GP) on existing code
- What has Genetic Improvement done
  - – Technology behind automatic bug fixing
  - – Improvement of existing code: speedup, transplanting, program adaptation
- Goals of Genetic Improvement
  - – more automated/higher level programming
  - – Build on open source, Infer test oracles
- 5 minute break
- Demo count blue pixels, unix,tsch,gcc

What is Genetic Improvement

Genetic Improvement

Use GP to evolve a population of computer programs

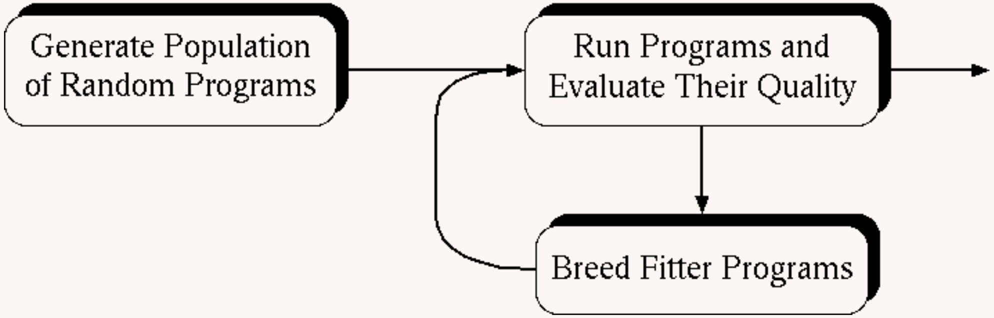
- Start with representation of human written code
- Programs' fitness is determined by running them
- Better programs are selected to be parents
- New generation of programs are created by randomly combining above average parents or by mutation.
- Repeat generations until solution found.



A Field Guide to Genetic Programming

Riccardo Poli
William B. Langdon
Nicholas F. McPhee

with contributions by
John R. Koza



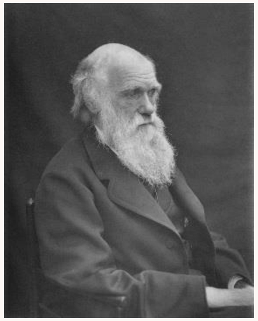
Generate Population of Random Programs
Run Programs and Evaluate Their Quality
Breed Fitter Programs

Solution

```
(IF (> x 15.43)
        (+ 2.3787 x)
        (SQRT y))
```

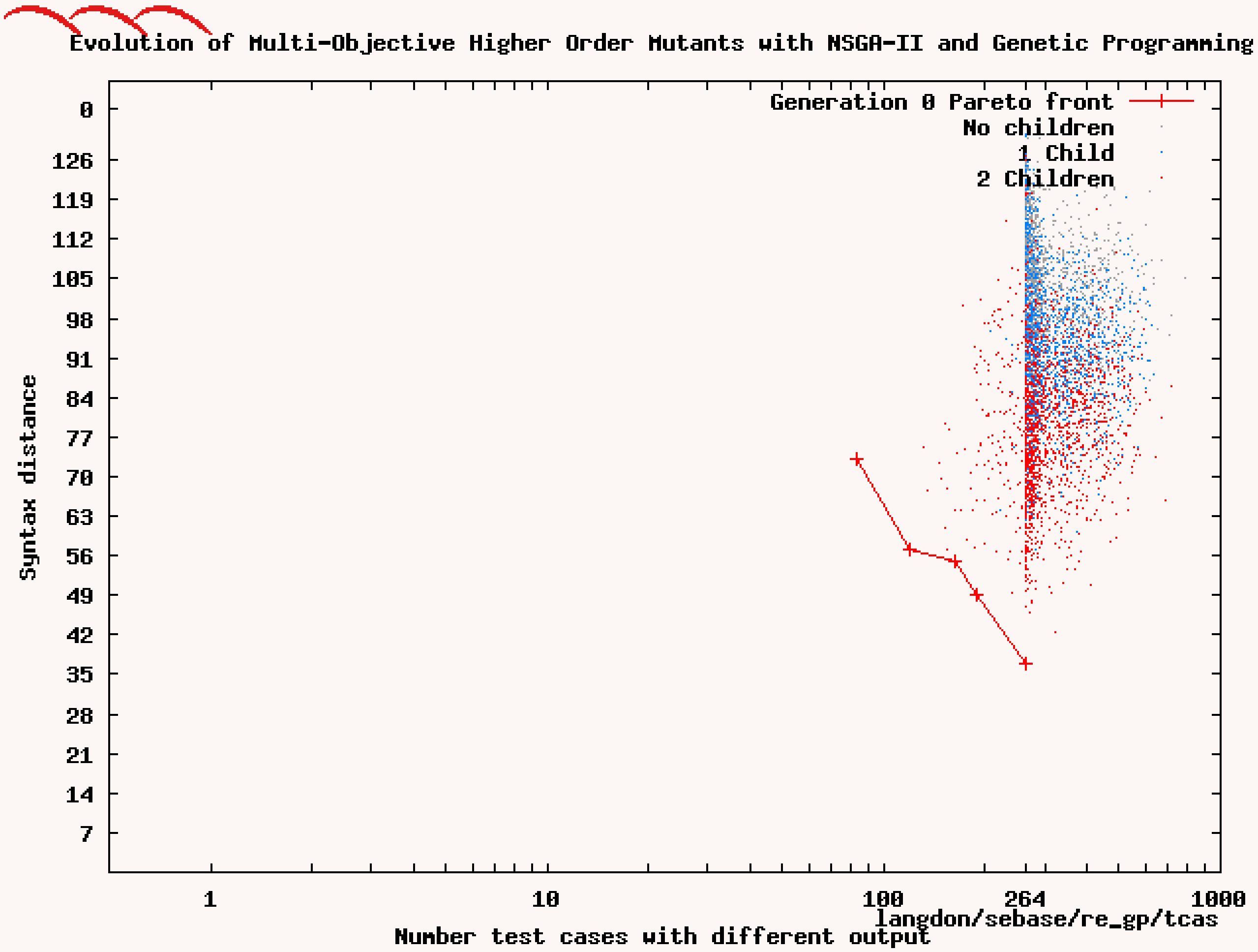
Free PDF

Free E-book kindle

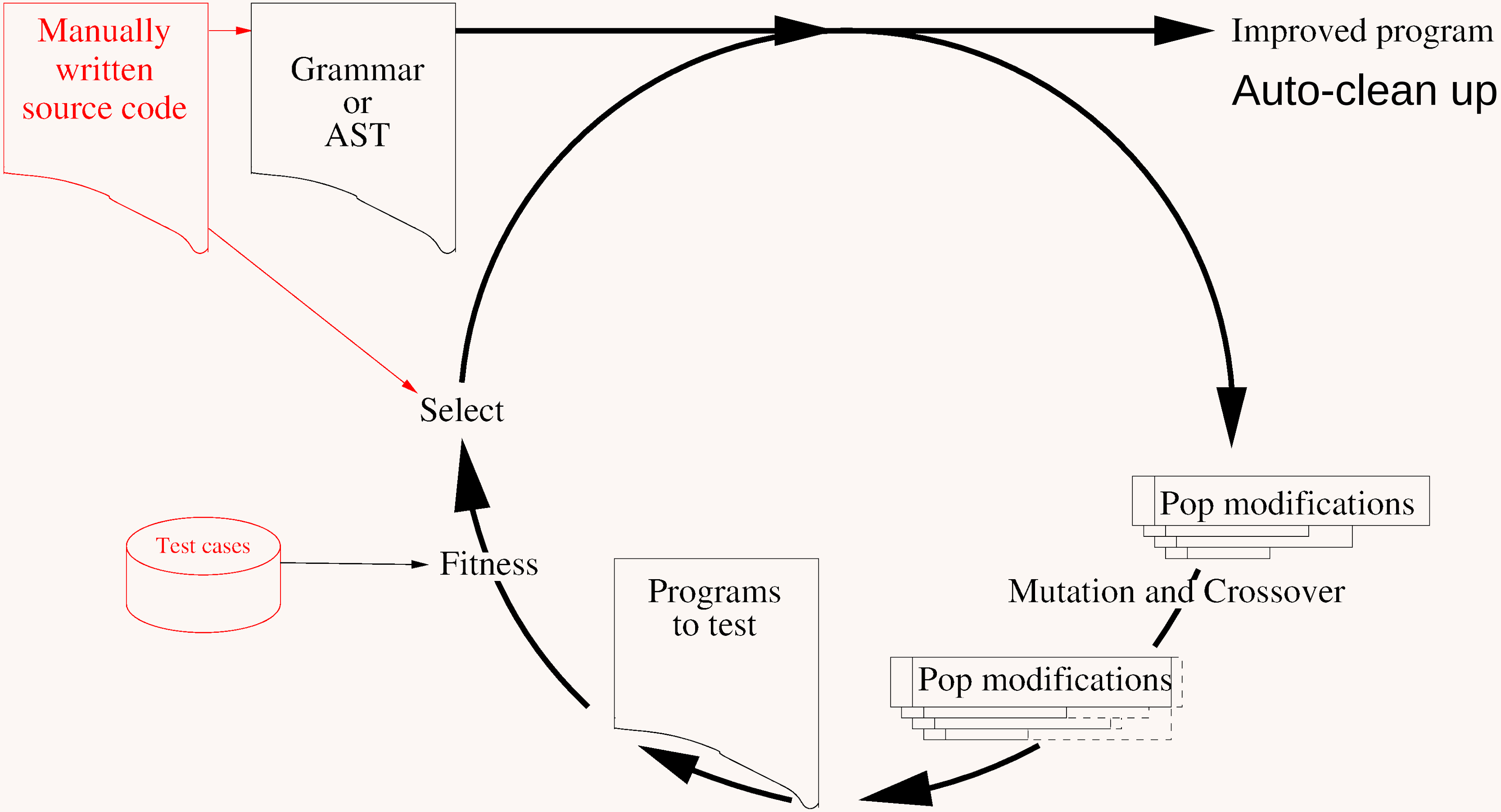


Charles Darwin 1809-1882

Evolving population of programs



Typical GI Evolutionary Cycle



Many types of mutation.
Eg replace line of C++ code with another from the same file.

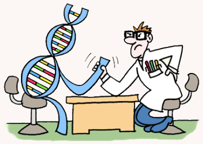
GI Automatic Coding

- Genetic Improvement does not start from zero
- Use existing system
  - ▪ Source of non-random code
  - – Use existing code as test “Oracle”. (Program is its own functional specification)
  - – Can always compare against previous version
  - – Easier to tell if better than if closer to poorly defined goal functionality.
- Testing scales (sort of). Hybrid with “proof” systems

What has
Genetic Improvement done

GP Automatic Bug Fixing (APR)

- Run code: example to reproduce bug, a few tests to show fixed code still works.
- Search for replacement C statement within program which fixes bug. Fault location tool
- Real bugs in real programs (mostly C/C++ or Java).
  - Multiple prizes and best papers, including:
  - 1st prize Human-Competitive [ICSE] Gold Humie



- In daily use: Iceland health clinic [GI-2017] Python

GI to Speed up human written programs

- Bowtie2, 70 times faster [IEEE TEVC 2015]
- GPGPU BarraCUDA [BioData Mining]
  - In use since 2015. 3000 downloads from SF
  - On real data speed up to 3 times (arXiv.org)
  - Commercial use by Lab7 (in BioBuilds 2015)
  - Ported by IBM to their Power8
  - Cambridge Epigenetix

GTX 1080 21x faster than bwameth (twin core CPU)

Microsoft Azure GPU cloud

Genetic Improvement to Reduce Resource Consumption

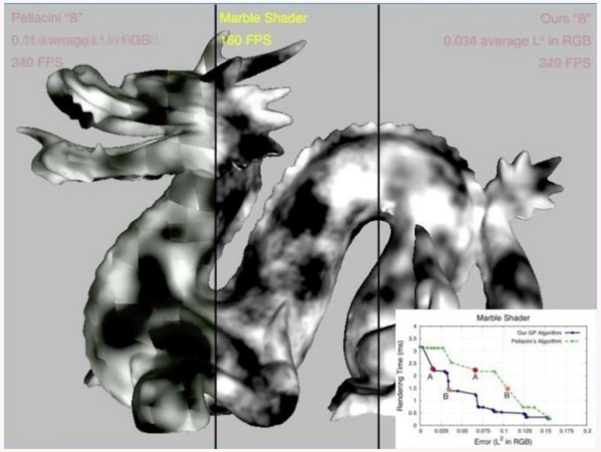
- Energy reduction [GECCO 2015a,SSBSE] particularly for mobile computing [GI-2017]
- RAM memory reduction [GECCO 2015b]
- Reduce run time [pknotsRG,OpenCV, RNAfold]
- Choose better library [SSBSE-2017]
- Improve library [SSBSE 2014,2016]

GI to Improve functionality

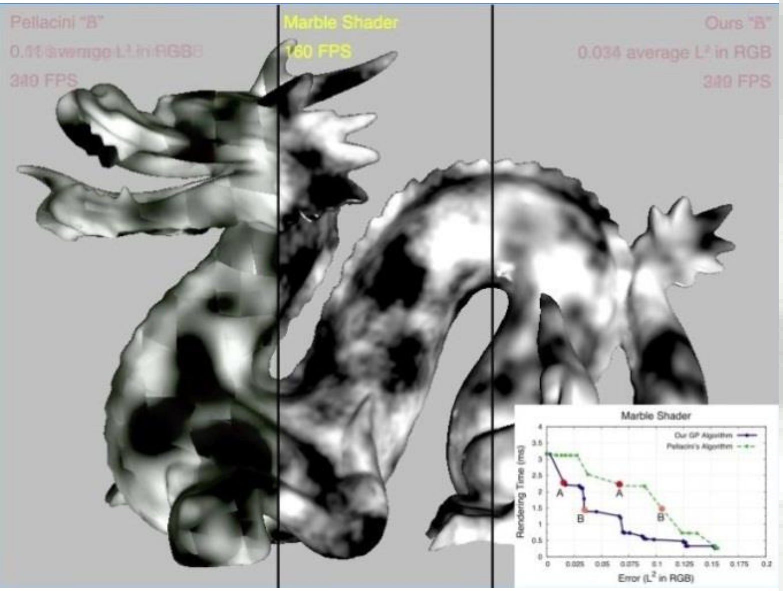
- Transplanting C++ [Marginean SSBSE'15, ISSTA'15]

  E.g. graph layout into Kate, H.264 into VLC, awarded Gold Humie, 26hours CPU v. 20days

- Autoporting
  - gzip to GPU [CEC 2010], RNAfold to SSE [GI-2017]
- Better RNA structure prediction
- Improving GPU shaders [2011]



GI Improving GPU shaders [2011]



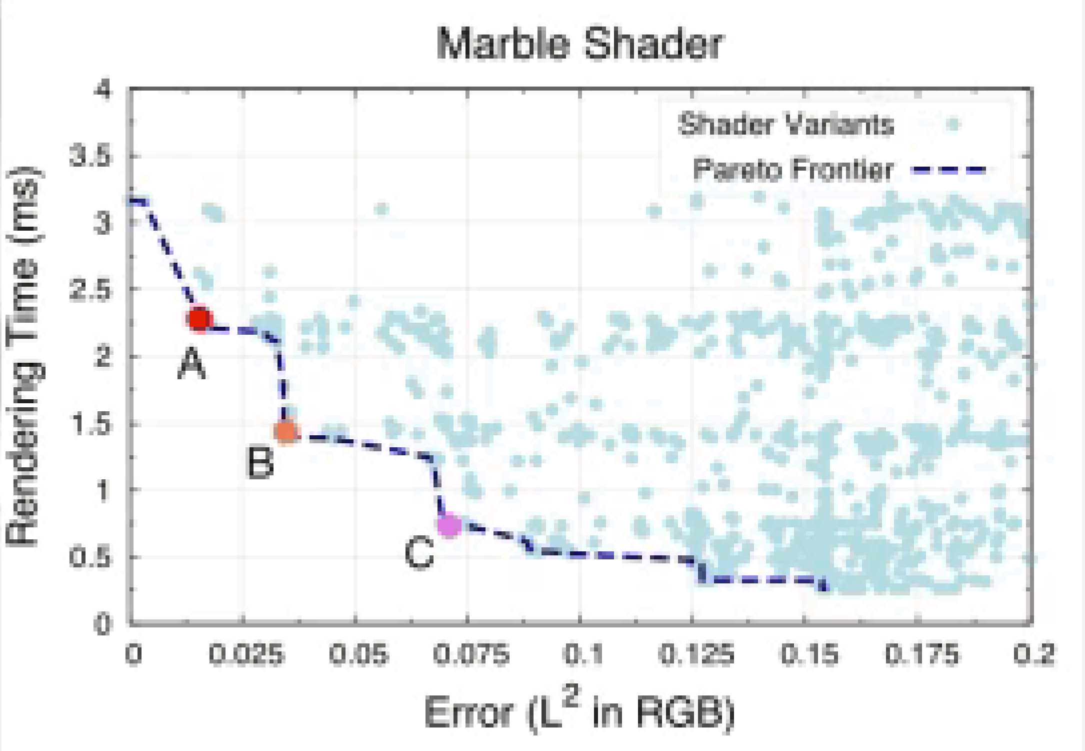


Fig 1. number of core papers on genetic improvement

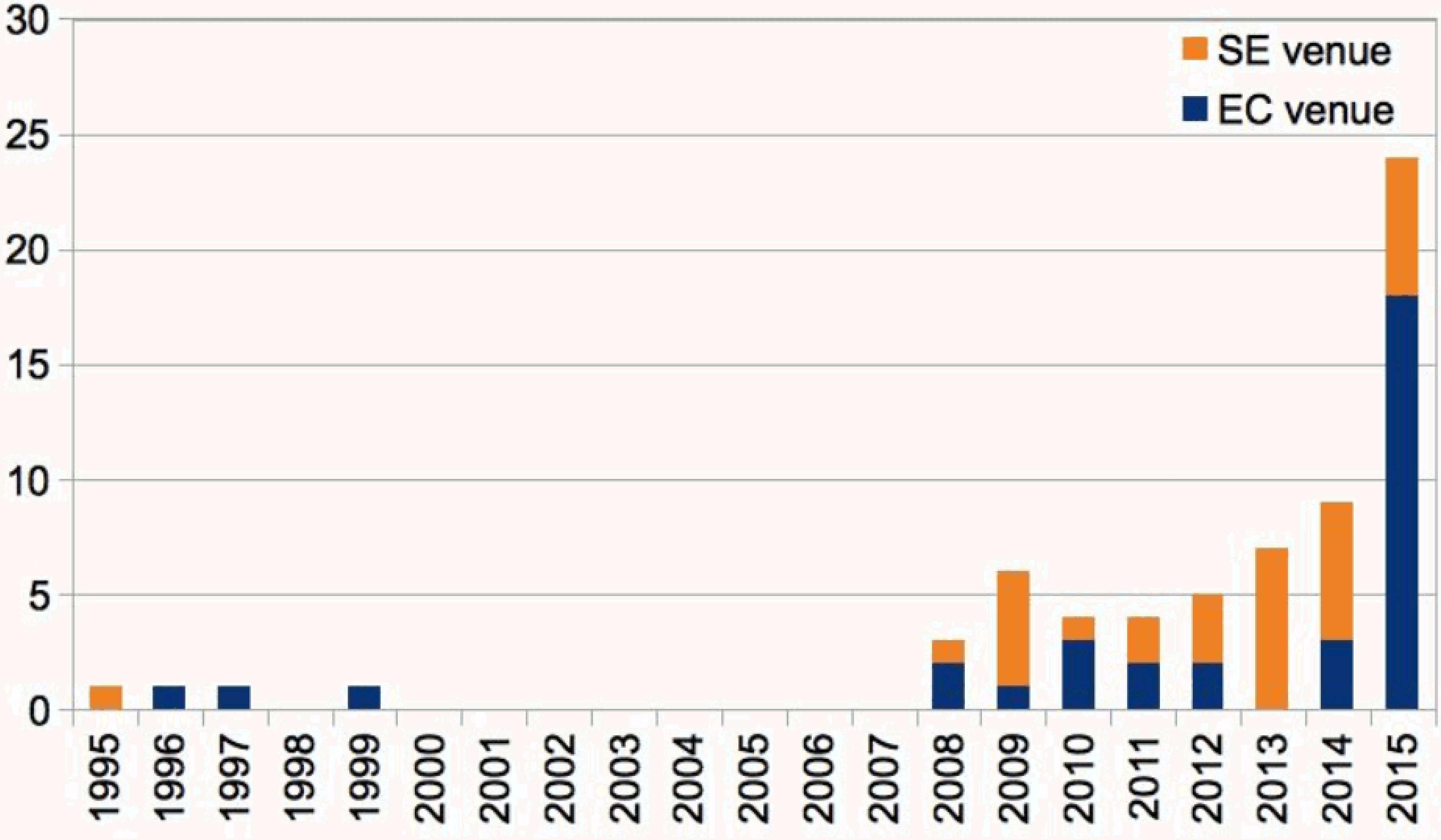
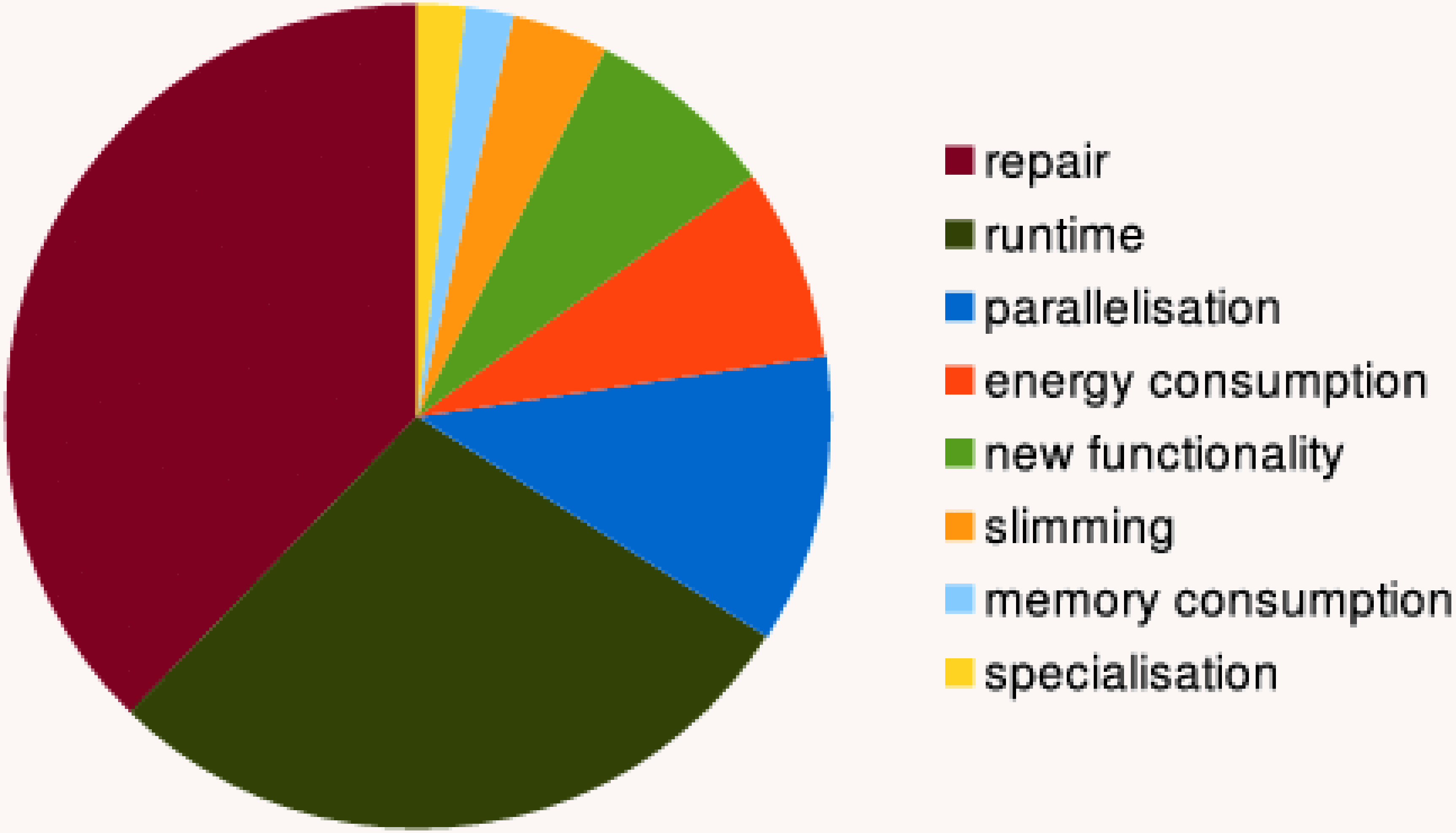


Fig 3. software applications of empirical studies in core papers on genetic improvement



Where is Genetic Improvement Headed

- Automatic coding
- Automatic testing (oracles)
- code transplant, data transplant

Goals of Genetic Improvement

- Totally automatic programming still distant
- Intermediate GI as stepping stone to higher level programming. Programmer says what needs to be done by test cases.
- Program assembled from existing(perhaps open source) code or more automated bag-of-parts software product lines (mashups)
- Automatic customisation, per user versions, many (30184 [Monperrus,2017]) version computing

GI Automatic Coding

- Genetic Improvement may also allow us to trade improvement in one aspect against loss in another.
  - E.g. reduce accuracy but faster execution
  - (Can sometimes improve both)
- Customise per user (dreaming smart phone)
- Predict what user will want to next.
  - E.g. yesterday read news page at 8:30 so today load it into cache before they reach underground tube station.

Automatic Testing

- Hardware Improvement: Tetsuya Higuchi Analogue EHW chip for mobile phones
- Software quality continues to be dominated by the cost of manual effort
- Existing test suites are often run automatically
- Evolution can automatically create test cases (goal: code coverage) but still lacks knowledge of the correct answer (known as the test oracle problem).

Automatic Oracle Generation

- Current automatic oracles are crude:
  - did the program terminate? Did it crash?
- Given huge number of existing open source test suites [SBST 2017], can Machine Learning:
  - infer the answer expected of a test case?
  - could Machine Learning get close or give plausible answers?
  - Reject non-plausible answers?

Automatic Data Transplant

- Tuning 50000 int in RNAfold to better fit empirical scientific data [EuroGP 2018].
  - Better predictions on average (some better, some worse) on known RNA structures.

- Converting a GNU C library sqrt double function to calculate cube root instead by using evolution to adjust 1024 floats [SSBSE 2018,RN18/05]
  - plus manual code tweaks

Six impossible things before breakfast

- To have impact do something considered impossible.
- If you believe software is fragile you will not only be wrong but shut out the possibility of mutating it into something better.
- Genetic Improvement has repeatedly shown mutation need not be disastrous and can lead to great things.

Conclusions

- Genetic Improvement (GI) applies Darwinian survival of the fitness to existing code
- GI for automatic bugfixing, software transplanting, performance improvement faster answers or better answers.

  BarraCUDA 3,095 sourceforge downloads (26 months).

  Commercial use by Lab7 (in BioBuilds Nov2015)

  IBM Power8.

  RNAfold par,SSE,CUDA (17,061 downloads)

- Future GI. Do impossible things
- Software is not fragile

  break it, bend it, Evolve it

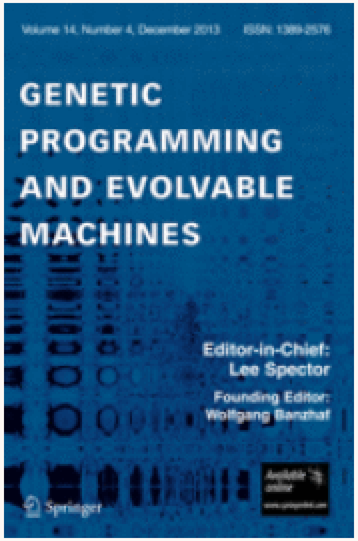
CREST

WIKIPEDIA

Genetic Improvement

GI 18

Five workshops on GI,
Dagstuhl seminar



Volume 14, Number 4, December 2013 ISSN: 1389-2576

GENETIC PROGRAMMING AND EVOLVABLE MACHINES

Editor-in-Chief: Lee Spector

Founding Editor: Wolfgang Banzhaf

Springer

GI special issue

DEMO in 5 minutes

Assumes Linux, tcsh, gcc

http://www.cs.ucl.ac.uk/staff/W.Langdon/ftp/gp-code/opencv_gp.tar.gz

See blue.cpp in README.txt

EPSRC

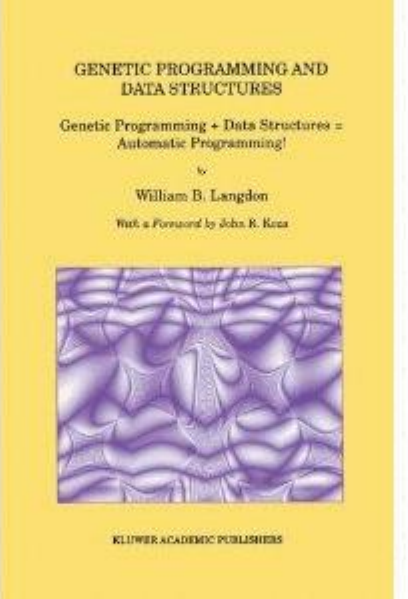
Genetic Improvement

UCL

W. B. Langdon

CREST

Department of Computer Science



GENETIC PROGRAMMING AND DATA STRUCTURES

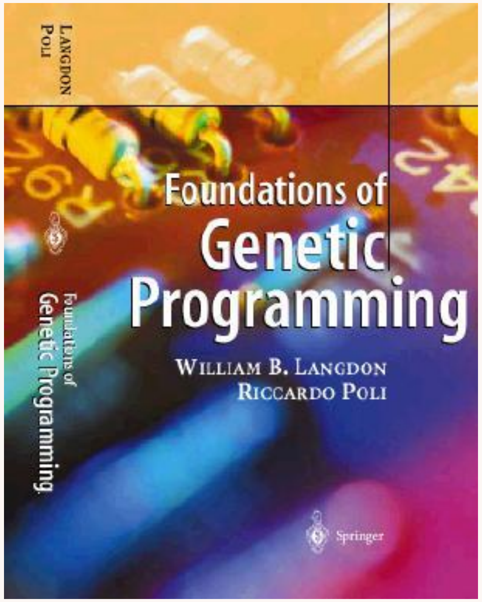
Genetic Programming + Data Structures = Automatic Programming!

by

William B. Langdon

With a Foreword by John R. Koza

KLUWER ACADEMIC PUBLISHERS



LANGDON POLI

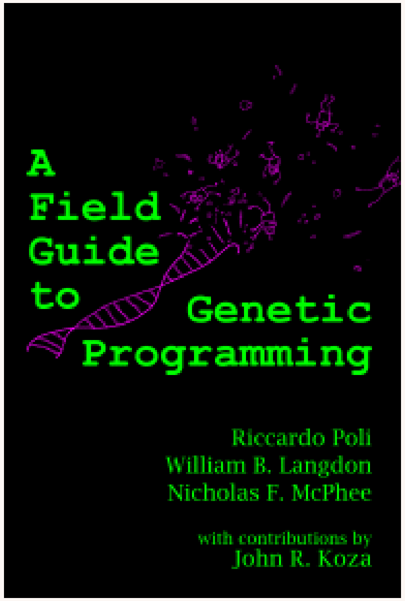
Foundations of Genetic Programming

Foundations of Genetic Programming

WILLIAM B. LANGDON

RICCARDO POLI

Springer



A Field Guide to Genetic Programming

Riccardo Poli

William B. Langdon

Nicholas F. McPhee

with contributions by

John R. Koza

Demo count blue pixels

- Assumes unix, tsch gcc
- http://www.cs.ucl.ac.uk/staff/W.Langdon/ftp/gp-code/opencv_gp.tar.gz
- gunzip -c ftp/gp-code/opencv_gp.tar.gz | tar xvf –
- README.txt

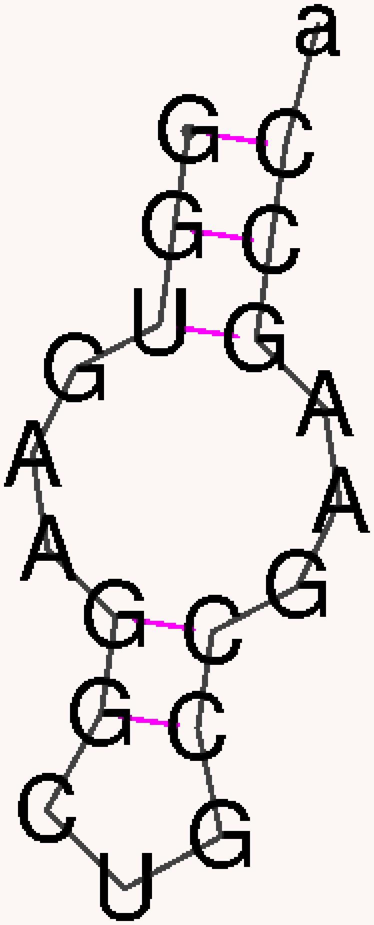
Genetic Improvement in Emergent and Self-adaptive Systems

- ECSELR adapt Java code via JVM
  - Evolution runs inside the Java Virtual Machine
  - Tune distributed www online video services in response to changes in user geographic load
- Adaptation: detect and repair buffer overruns [failure-oblivious computing]

  (Faster to trap invalid memory exception than test every access for end-of-buffer [ukmac 2016])

GI and Dynamic Programming

- Use evolution to adapt 50,000 parameters of dynamic programming model.
- RNAfold predicts RNA structures based on finding one with minimum energy.
- RNA is a long chain biomolecule (cf. DNA) which reduces energy by binding to self.
- Adjust parameters to fit reality



Big Data and Self Adaptation?

- Deep learning needs large data, so
- What are suitable data open sources of training data to aid *adaptation* e.g. to user

1. User feedback
   - Sparse, badly structured? mobile app stores?
   - 5% users are testers, instrument user
   - Dreaming smart phones
2. Bug reports/crash dumps
   - Bugzilla 200 per week (cf. Iceland health clinic)
   - Bug triage tries to discard lots of data. Can we learn from it instead?

BNF Grammar

Configuration parameter

```
if (*lastpos!=pos_shifted)
{
#ifndef sequence_global
  *data = tmp = tex1Dfetch(sequences_array, pos_shifted);
#else
  *data = tmp = Global_sequences(global_sequences,pos_shifted);
#endif /*sequence_global*/
  *lastpos=pos_shifted;
}
```

CUDA lines 119-127

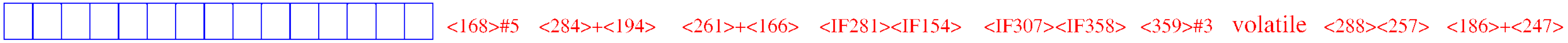
```
<119>   ::= " if" <IF_119> " \n"
<IF_119>::= "(*lastpos!=pos_shifted)"
<120>   ::= "{\n"
<121>   ::= "#ifndef sequence_global\n"
<122>   ::= "" <_122> "\n"
<_122>  ::= "*data = tmp = tex1Dfetch(sequences_array, pos_shifted);"
<123>   ::= "#else\n"
<124>   ::= "" <_124> "\n"
<_124>  ::= "*data = tmp = Global_sequences(global_sequences,pos_shifted);"
<125>   ::= "#endif\n"
<126>   ::= "" <_126> "\n"
<_126>  ::= "*lastpos=pos_shifted;"
<127>   ::= "}\n"
```

Fragment of Grammar (Total 773 rules)

Grammar rule types

- Type indicated by rule name
- Replace rule only by another of same type
- Fixed (evolution cannot change code), variable.
- statements (e.g. assignment, **Not** declaration)
- <_947> ::= "*k0 = k;"
- IF
- <_392>   ::= " if" <IF_392> " {\n"
- <IF_392>   ::= " (par==0)"
- for loops (for1, for2, for3)
- "for(" <for1_630> ";" <for2_630> ";" <for3_630> ") \n"
- ELSE
- specials

Representation



- variable length list of grammar patches.
  - no size limit, so search space is infinite
- tree like 2pt crossover.
- Mutation adds one randomly chosen grammar change
- 3 possible grammar changes:
  - Delete line of source code (or replace by “”, 0)
  - Replace with line of GPU code (same type)
  - Insert a copy of another line of kernel code

Example Mutating Grammar

```
<_947> ::= "*k0 = k;"
<_929> ::= "((int*)l0)[1] =
__shfl(((int*)&l)[1],threads_per_sequence/2,threads_per_sequence);
"
```

2 lines from grammar

```
<_947>+<_929>
```

Fragment of list of mutations

Says insert copy of line 929 before line 947

Copy of line 929

New code

```
((int*)l0)[1] =
__shfl(((int*)&l)[1],threads_per_sequence/2,threads_per_sequence);
*k0 = k;
```

Line 947

The Genetic Programming Bibliography

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

12259 references, 10000 authors

Make sure it has all of your papers!
E.g. email W.Langdon@cs.ucl.ac.uk or use | Add to It | web link

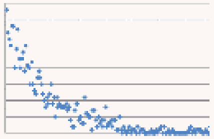
Co-authorships

RSS Support available through the Collection of CS Bibliographies.

Downloads by day

Co-authorship community. Downloads

A personalised list of every author's GP publications.



Your papers

blog

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