Queen Mary University of London, 12th December 2018

Genetic Improvement by Evolving Program Data

W. B. Langdon

Computer Science, University College London

A Comprehensive Survey, IEEE TEVC

GI 2019

GI 2019, Montreal, ICSE-2019 workshop

Simple blue example of Genetic Improvement

opencv_gp.tar.gz

RN/18/06

12.12.2018 (short)

Wikipedia Genetic Improvement

Genetic Improvement of Software

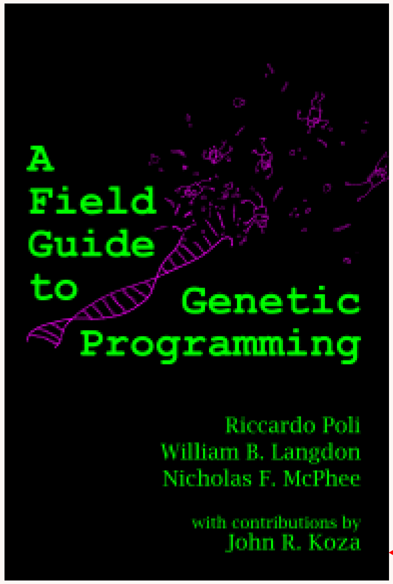
- What is Genetic Improvement
  - Genetic Programming (GP) on existing code
- What has Genetic Improvement done
  - Technology behind automatic bug fixing
  - Improvement of existing code: speedup, transplanting, program adaptation, parallel, mobile energy reduction
- Genetic Improvement of Data
  - RNAfold (GI parameters shipped since 2.4.7)
  - Transforming GNU C library: cbrt, log2
- Conclusions

What is Genetic Improvement

Genetic Improvement

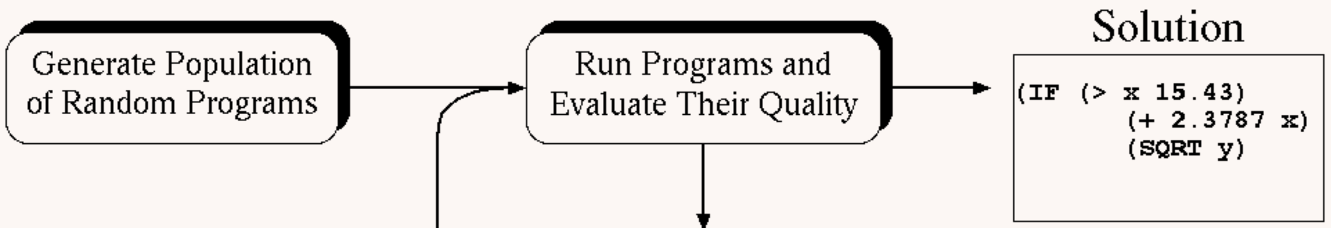
Use GP to evolve a population of computer programs

- 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.

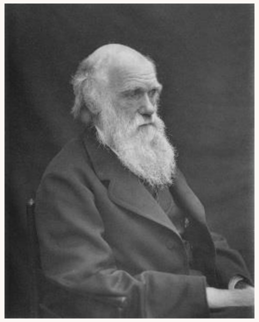


Free PDF

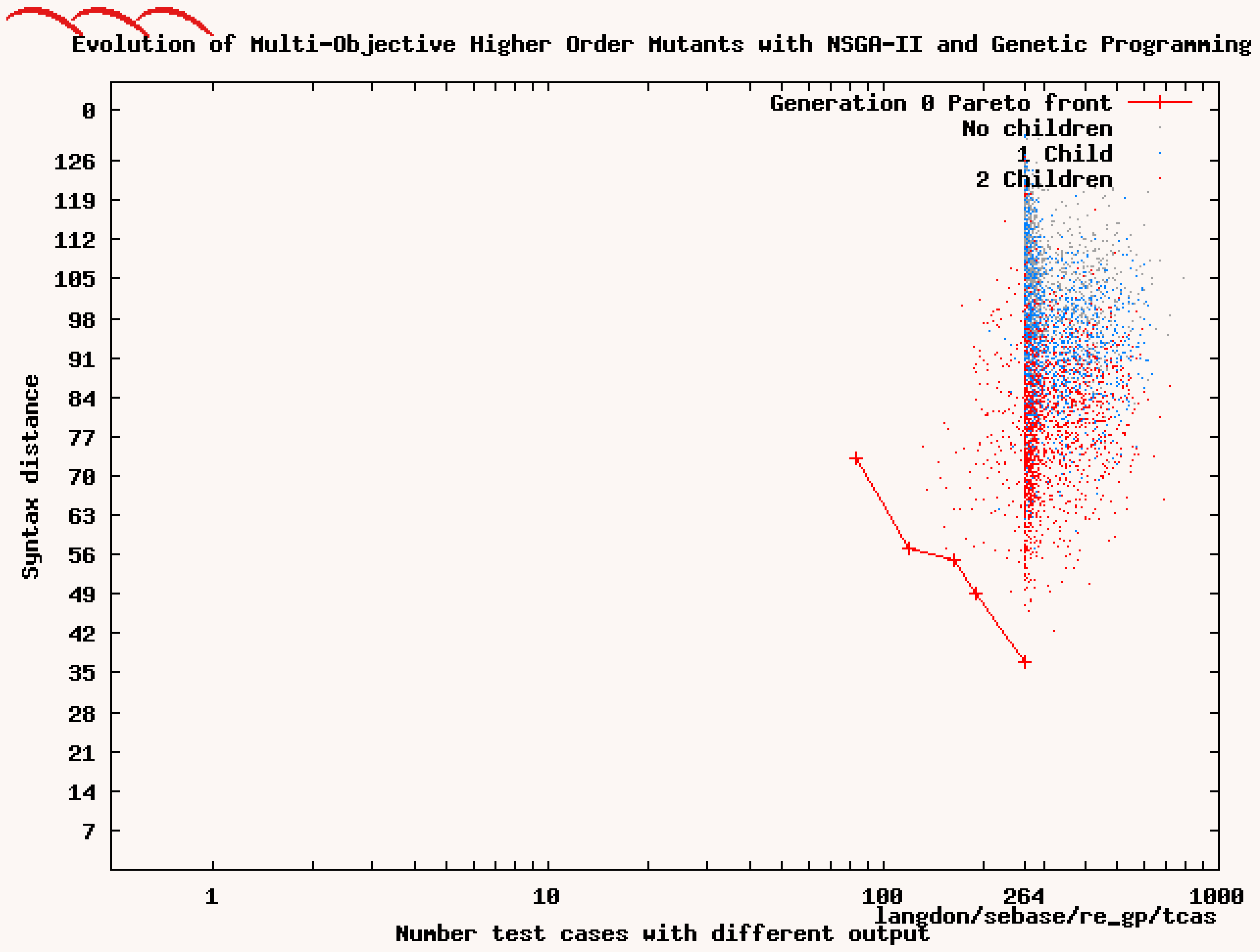
Free E-book kindle


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

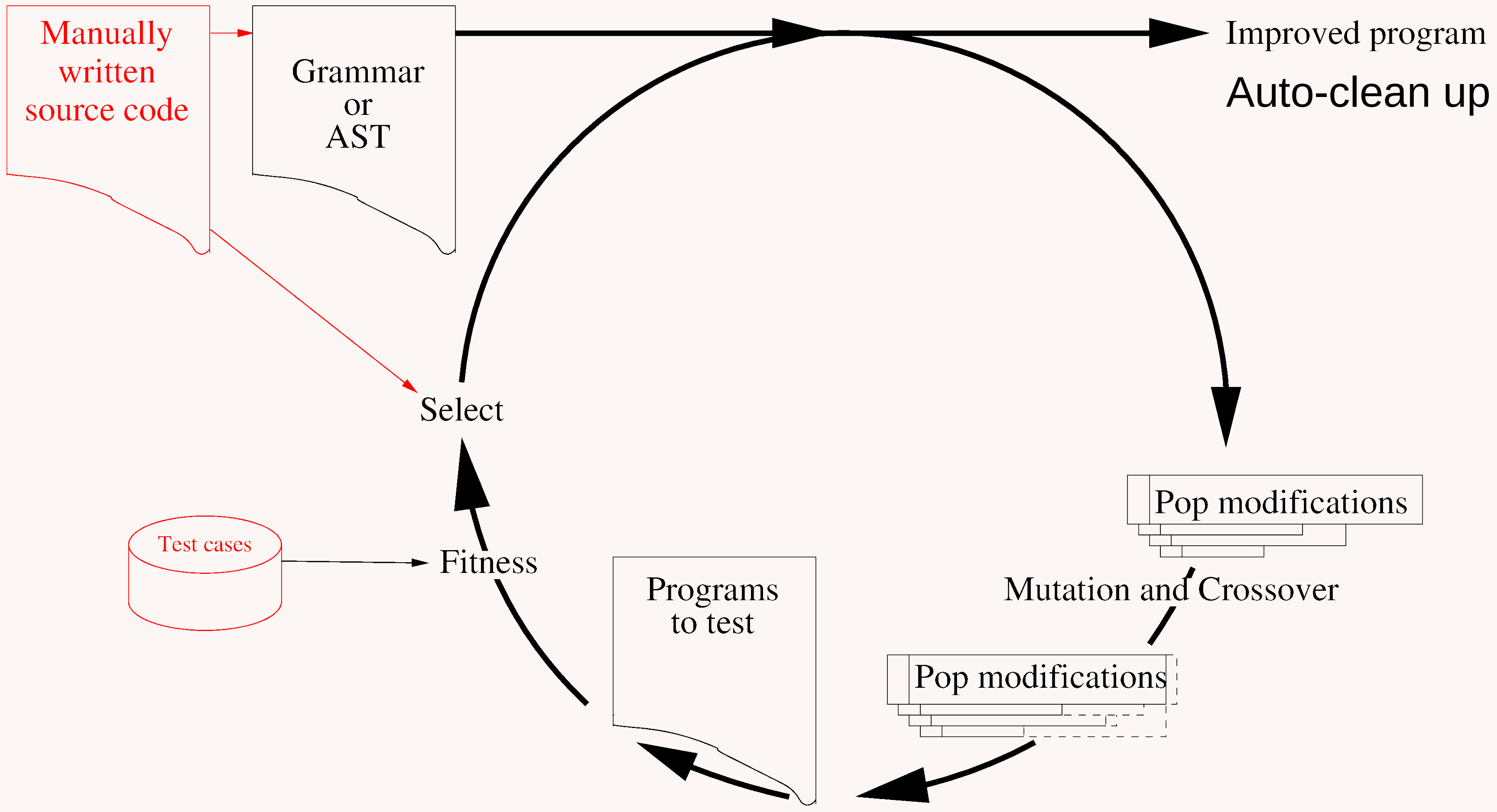
```
(IF (> x 15.43)
        (+ 2.3787 x)
        (SQRT y)
```


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

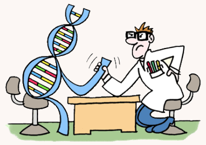
- Genetic Improvement does not start from zero
- Use existing system
  - Source of non-random code
  - Use existing code as test “Oracle”. (Program is its own functional specification)
  - Can always compare against previous version
  - Easier to tell if better than if closer to poorly defined goal functionality.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

What has Genetic Improvement done

CREST

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 Facebook SapFix Mark Harman



JANUS ENDURHÆFING EHF

C R E S T

GI to Speed up human written programs

- Bowtie2, 70 times faster [IEEE TEVC 2015]
- GPGPU BarraCUDA [BioData Mining]
  - In use since 2015. 4000 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

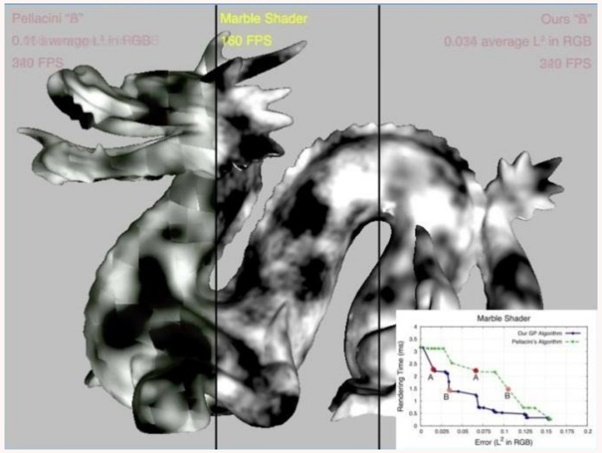
Genetic Improvement to Reduce Resource Consumption

- 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]

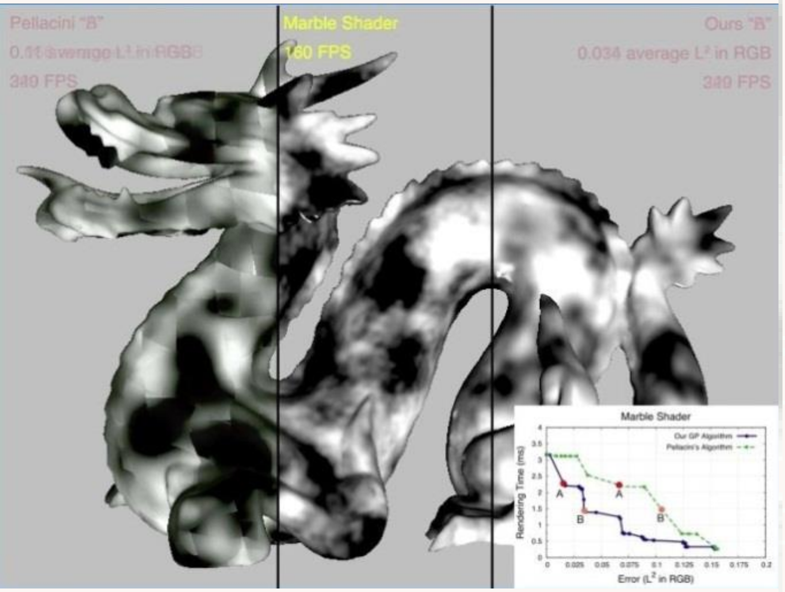
GI to Improve functionality

- 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]



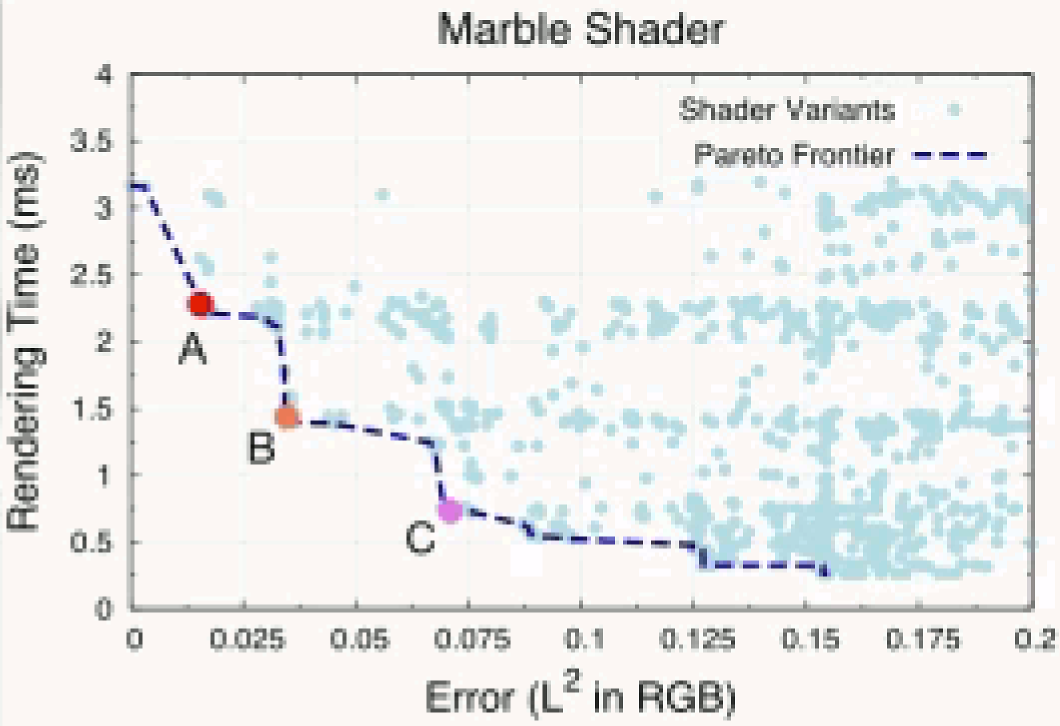


Fig 1. number of core papers on genetic improvement

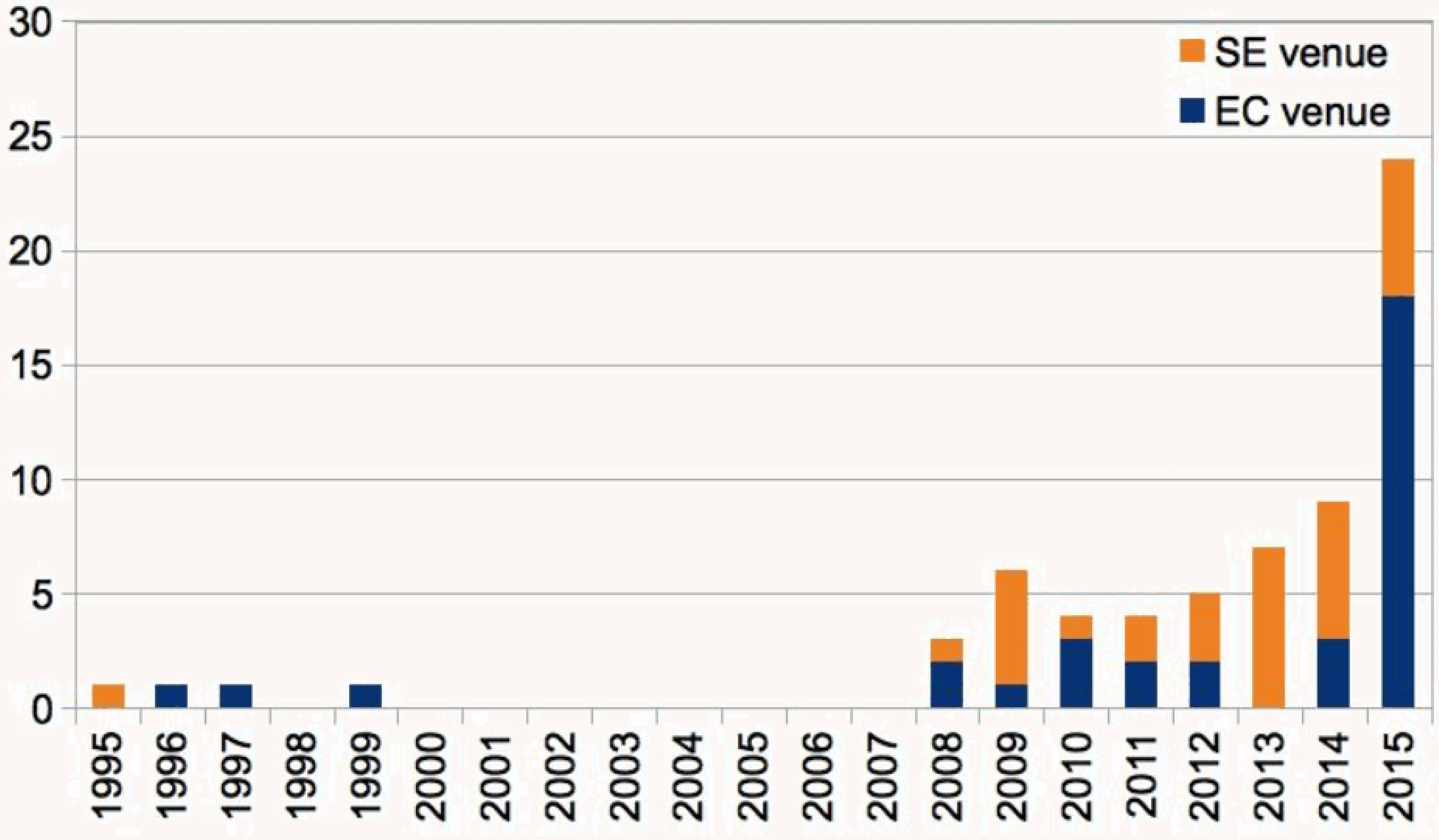
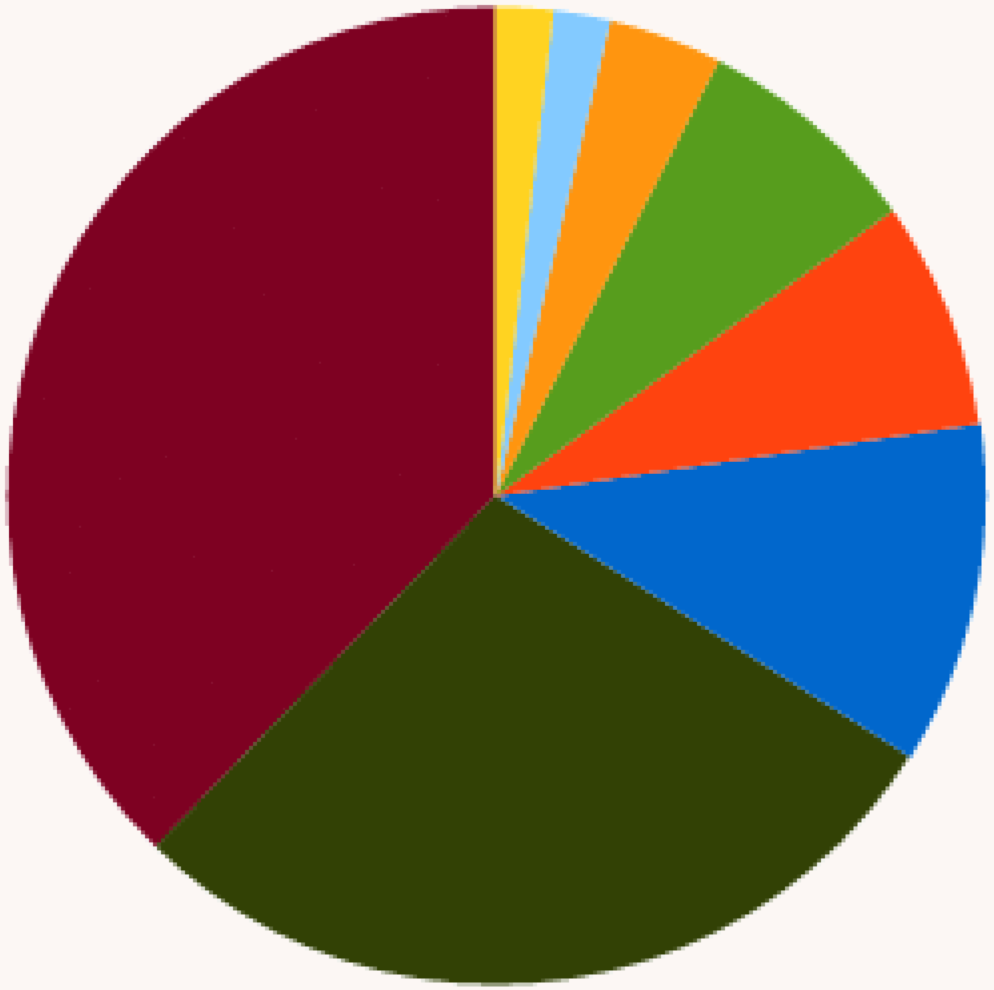
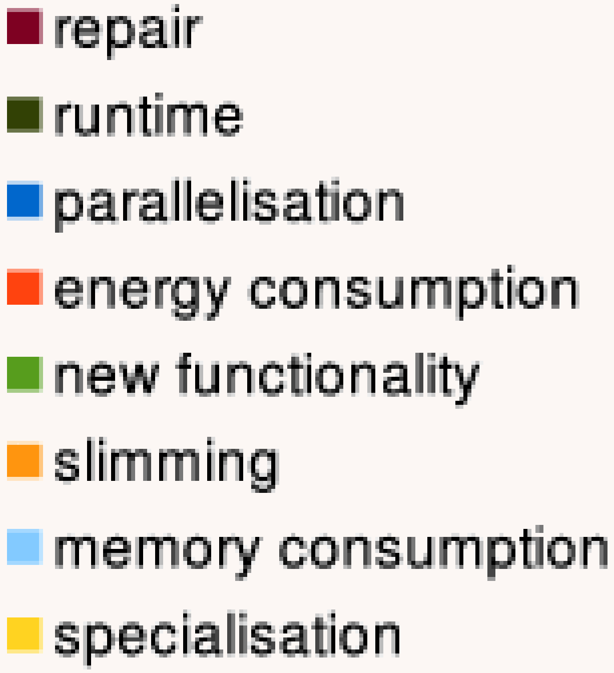


Fig 3. software applications of empirical studies in core papers on genetic improvement





Maintaining Embedded Constants

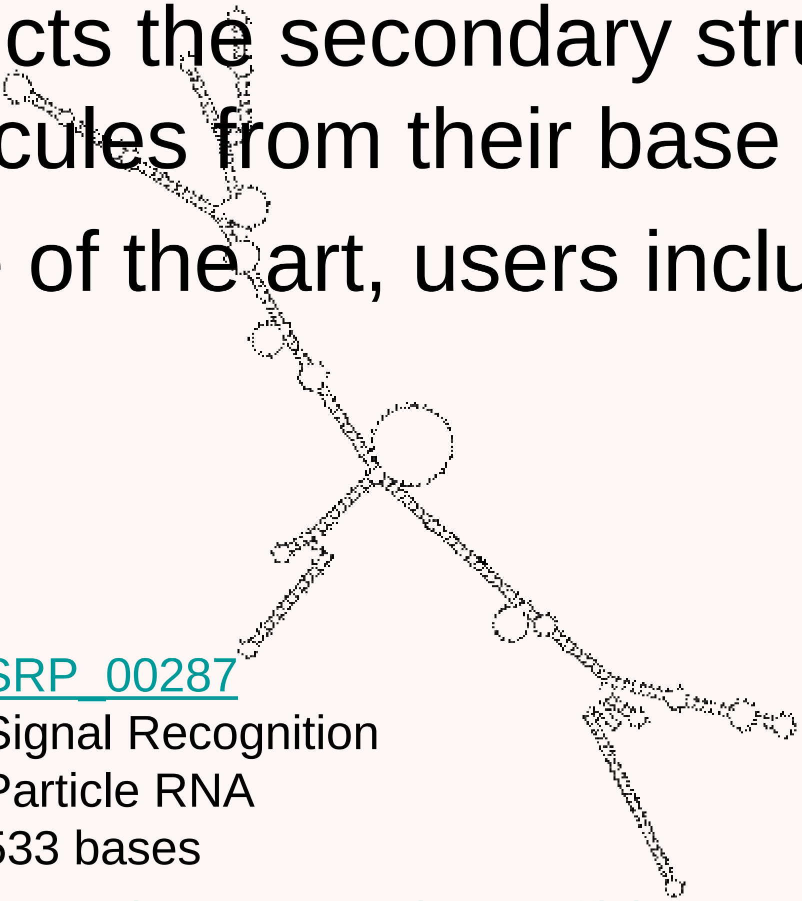
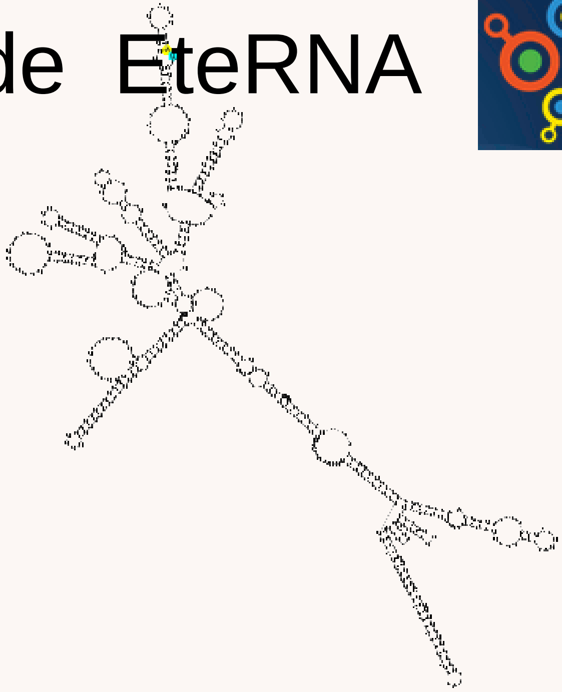
- EuroGP 2018
  - RNAfold 7000 lines of code 50000 numbers
  - net 20% better prediction of RNA structures
  - Shipped since 2.4.7
- SSBSE-2018 sqrt converted to cube root
- RN/18/05 generate $\log_2$ from open source maths framework

Genetic Improvement of RNAfold

- RNA→protein, enzyme, gene regulation
- Biomolecule shape controls function
- What is RNAfold
- Known RNA structures: RNA_STRAND
  – Evolving functional improvement
  – Representation
  – Mutation and Crossover
  – Fitness
- Evolved parameters better than published

What is RNAfold?

- Part of ViennaRNA package (170,000 lines)
- RNAfold 7100 lines .c (i.e. excluding .h)
- Predicts the secondary structure of RNA molecules from their base sequence
- State of the art, users include EteRNA

SRP_00287
Signal Recognition
Particle RNA
533 bases

Matthews correlation coefficient MCC 0.519169

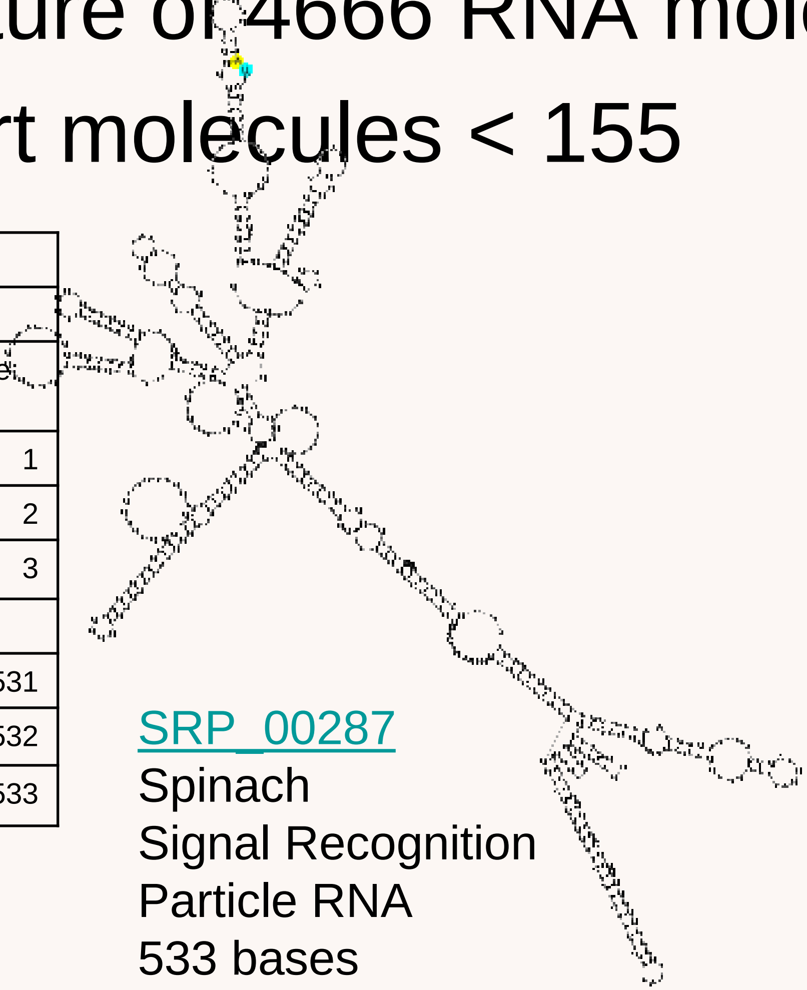
Training/Test data: RNA STRAND

Known structure of 4666 RNA molecules

Train on short molecules < 155

# File SRP_00287.ct					
# RNA SSTRAND database					
# External source: SRP Database, file name: SAC.CAS..ct, ID: SAC.CAS.					
1	A	0	2	15	1
2	G	1	3	14	2
3	G	2	4	13	3
…					
531	A	530	532	0	531
532	C	531	533	0	532
533	U	532	534	0	533

Paired positions

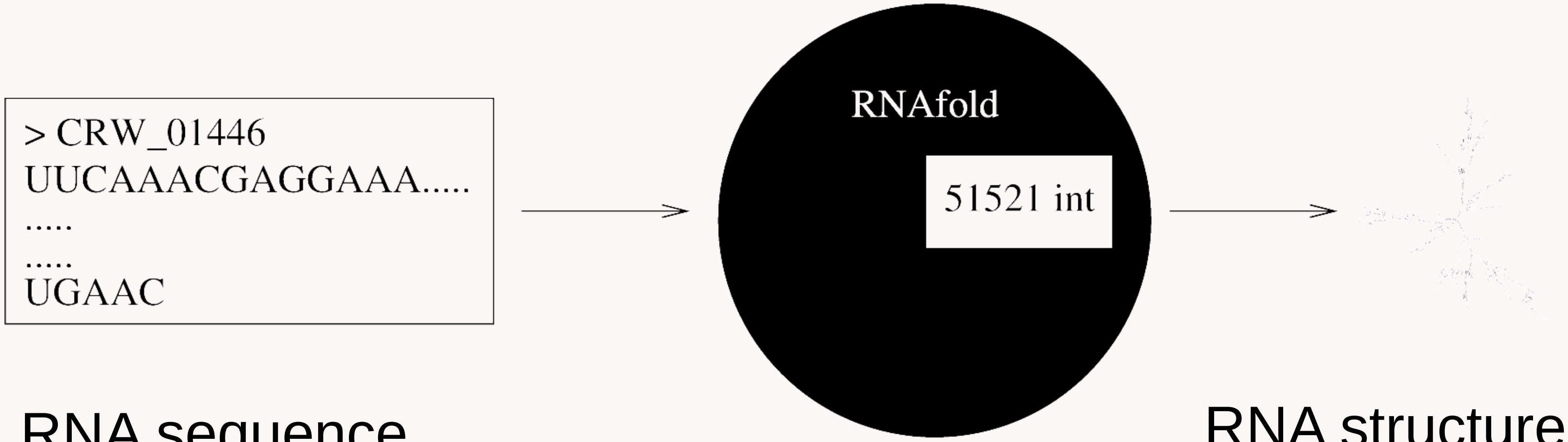


SRP_00287
Spinach
Signal Recognition
Particle RNA
533 bases

RNAfold

- Uses dynamic programming to select structure with minimum energy.
- Source code contains 31 read only scalars and arrays which hold parameters for model of interactions between RNA bases.
- Total 51745 parameters (all int)
- Use GP to optimise 51745 parameters

RNAfold



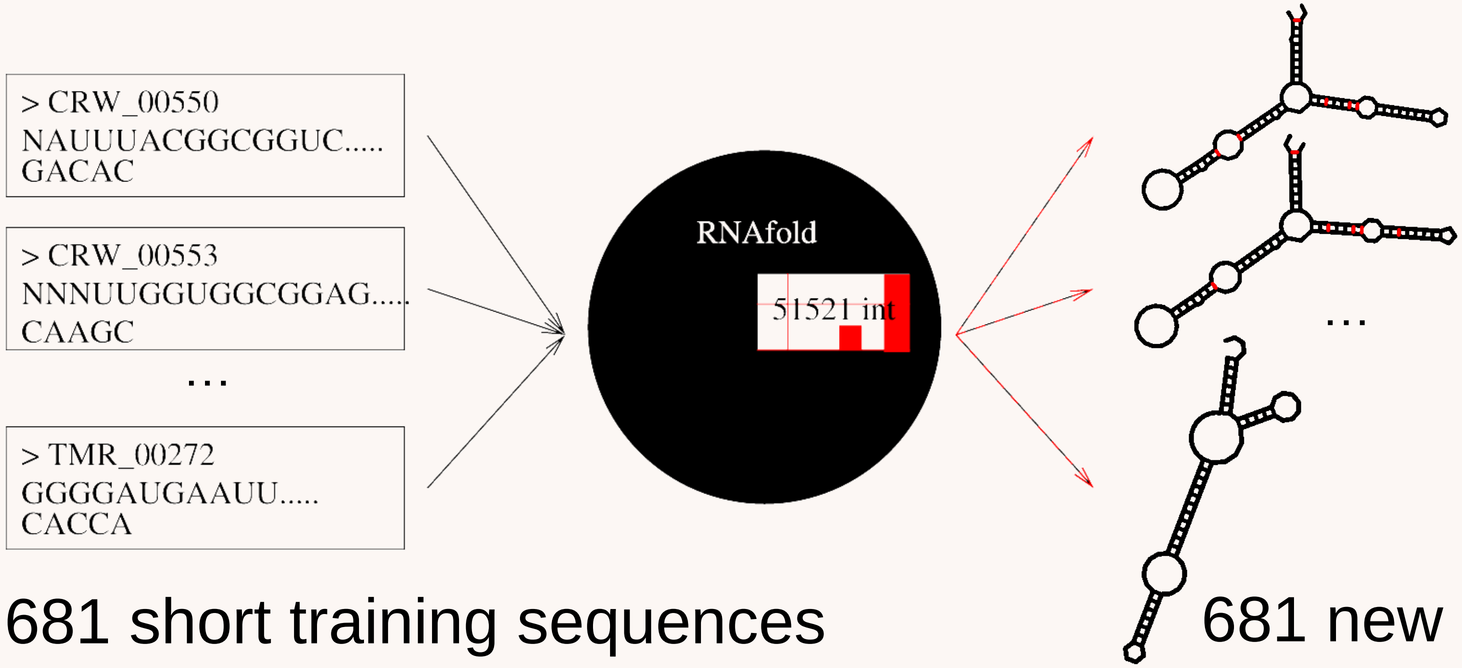
RNA sequence

RNA structure

RNAfold reads RNA molecules base sequence.
Outputs prediction of how molecule will fold up.
Internally RNAfold uses 51521 parameters.

CREST

Fitness of Mutated RNAfold



681 short training sequences

681 new predictions

- Mutate constants (file wbl_patch_arrays.txt)
- Run RNAfold on training RNA sequences
- Compare each new prediction with real structure
- Fitness mean Matthew’s correlation coefficient on 681 training RNA molecules

Optimise 50,000 parameters in RNAfold

- Mutate read-only arrays before RNAfold runs dynamic programming
- Compare new predicted structure with correct structure from RNA_STRAND
- Use ⅓ molecules for training
- Run time excessive:
  - use small molecules for training, size < 155
  - still running RNAfold 681 times (too many?)

Representation: Genotype→Phenotype

- Variable length genotype. Each gene specifies one or more changes to one scalar or array parameter.
- Apply changes in order (canonical operator removes some redundant genes, bloats anyway).
- Multiple types of mutation
- Two point (variable length) crossover

Mutate scalar or array values

- > Replace all values with another

  `int22 260>80` Replace every 260 with 80

- < Replace one or more values with another

  `mismatchI *,*,0<100` Volume overwrite [*,*,0] by 100

- Increment one or more values with another

  `mismatchM *,3,*+=20` Add 20 to all `mismatchM[*,3,*]`(40)

- Respect energy values (all multiples of 10 or INF) and “small values” (0…8). Cannot inc/dec INFinity.
- 20% creep mutation: change value in existing mutation.

Example individual

fitness 0.685112 changes 2387

noLP 2|dangle3 -10>-20|mismatch23I 0>70|hairpin *<530|int11 120>270|
mismatchM *,0,*+=-50

Variable length, e.g. six elements, separated by vertical bars

Data Structure	Operation	Effect
noLP	2	Set scalar noLP to 2
dangle3	-10>-20	All elements of dangle3 which are -10 set to -20
mismatch23I	0>70	All elements of mismatch23I which are zero set to 70
hairpin	*<530	Every element of hairpin set to 530
int11	120>270	All elements with value 120 replaced by 270
mismatchM	*,0,*+= -50	All elements in two slices (1st , 3rd dimension) are increment by -50

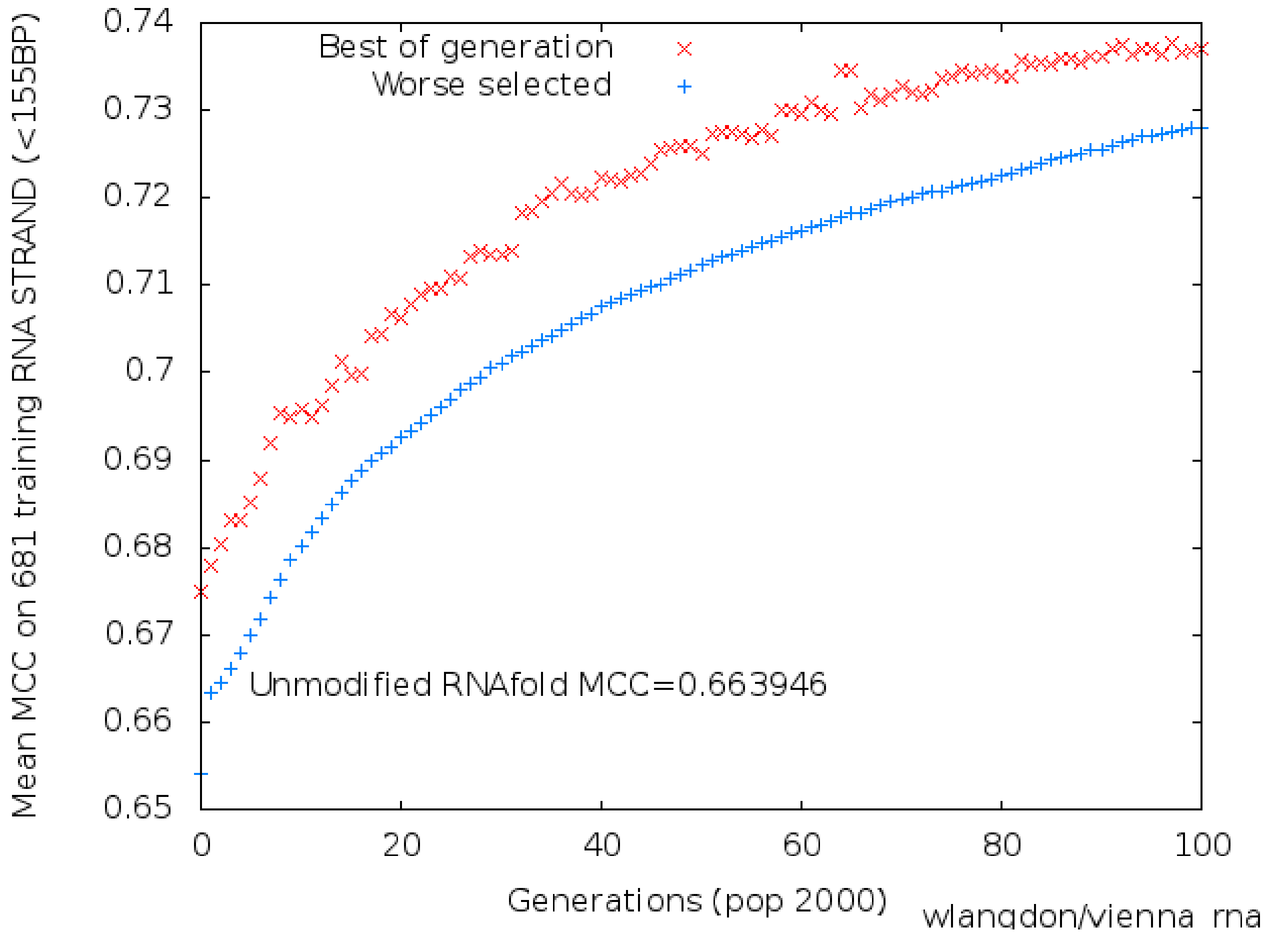
Fitness

- Run RNAfold on whole of training set of RNA molecules (len<155) from RNA_STRAND
- Compare each predicted structure against actual structure in RNA_STRAND using Matthews Correlation Coefficient (MCC) and against unmuted prediction. Fitness is mean MCC, but
  - If no changes: cannot be parent
  - If RNAfold segfaults: cannot be parent
  - If can’t mutate params: cannot be parent
- Select best half of population to be parents

Evolution

- 50% mutation, 50% crossover
- Promote search:
  - Reduce to canonical form
  - Tabu search to prevent repeated evaluation of genetically identical children
  - Anti-elitism: fitness cannot be parent more than 20 times (ie 1% popsize).
- 100 generations, population 2000

Evolution of Training Fitness



Results

- Take best of last generation (100)
  - Length 2849, MCC 0.737044
- Remove bloat by removing genes which do not help (two passes).
  - Length 42, MCC 0.737752
- Little over fitting: holdout MCC 0.730137

Evolved change

hairpin *<560
mismatchM -70>-130| *,3,*+=20| *,1,*+=-40| -110>-130| *,0,*+=-170| -60>-40
internal_loop *+=-40
MLintern *+=10| 3<-150
rtype 6<6| 2+=1
int11 *,*,*,*<200| 6,*,*,2+=-70
int21 230>260| *,*,*,*,3+=-70| 220>10000000
int22 260>80| 180>280| *,*,2,*,*,*+=10| 280>200| 200>10000000
dangle3 5,*+=-80
mismatchH *,*,*+=-90| *,*,3<-130| *,1,2<-80
mismatchExt *,*,*+=80| *,*,1<-40
TerminalAU 80
mismatch23I 70>10000000
mismatchI *,*,0<100| *,*,1+=-10| 2,3,1+=-100| *,4,*+=-40
ninio[2] 80
dangle5 *,*+=60
stack -100>60| -140>0| 2,2+=-20| *,4<-50
mismatch1nI 70>110
bulge *+=40

mismatchM many changes

rtype base A treated as C, X as K

mismatchH Rewrite array

mismatchI many changes

stack many changes

Impact on MCC

mismatch1nI	0.47%
mismatch23I	0.64%
int22	1.11%
dangle3	1.86%
int21	4.12%
dangle5	4.43%
bulge	5.15%
TerminalAU	6.02%
ninio[2]	7.53%
int11	10.70%
MLintern	10.72%
internal_loop	10.89%
hairpin	10.97%
mismatchExt	15.45%
stack	20.32%
mismatchI	21.12%
rtype	21.48%
mismatchM	21.62%
mismatchH	27.91%

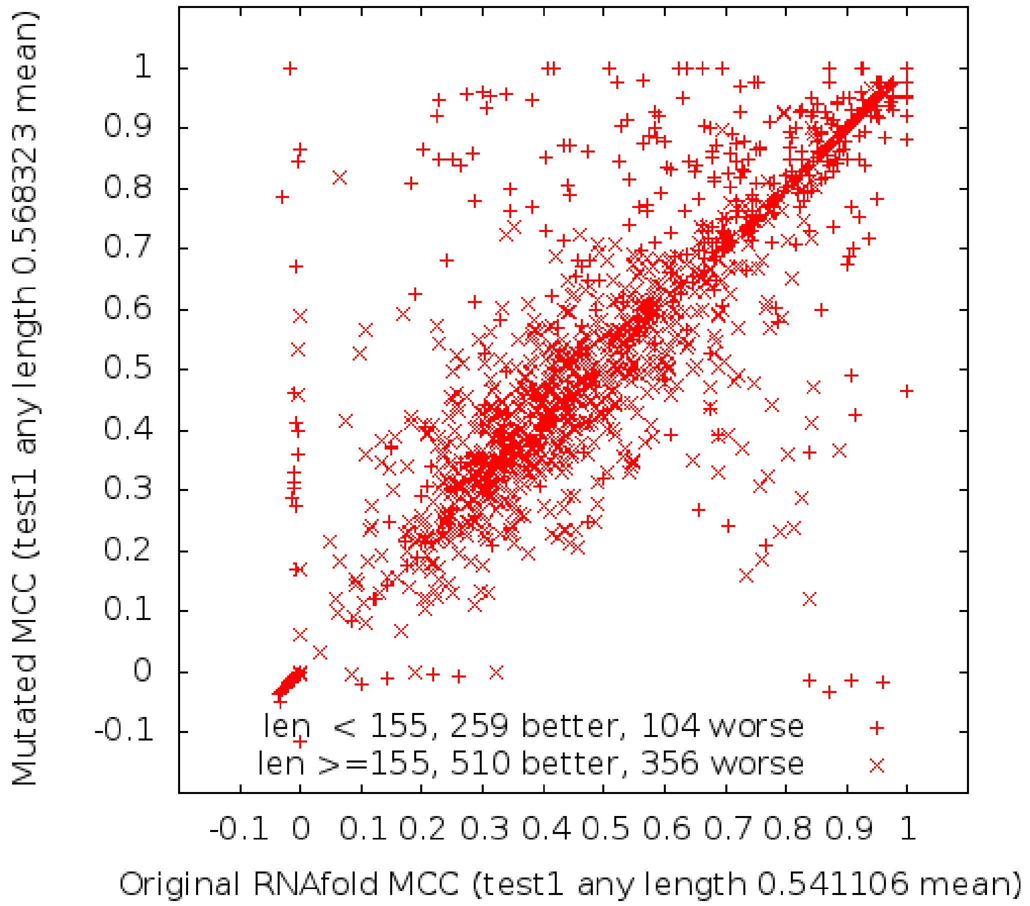
Fraction of improvement in MCC lost if remove changes to each scalar or array.
(Measured on training data.)

CREST

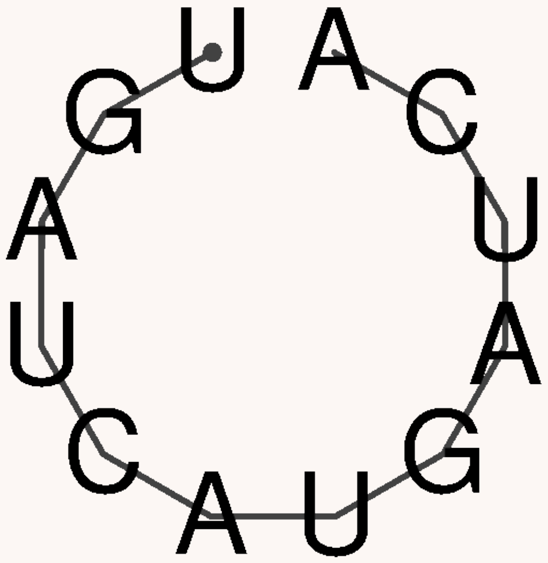
Out of Sample Performance

Generalises (MCC on test set ≈ training) and extrapolates (MCC long RNA similar to training).

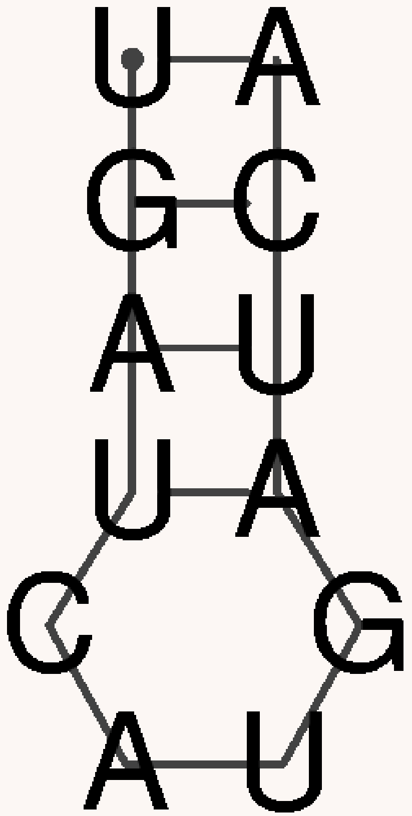
Total 769 better, 460 worse on holdout ⅓ RNA STRAND (1553).
Total overall out-of-sample improvement 19.897%



NDB_00028

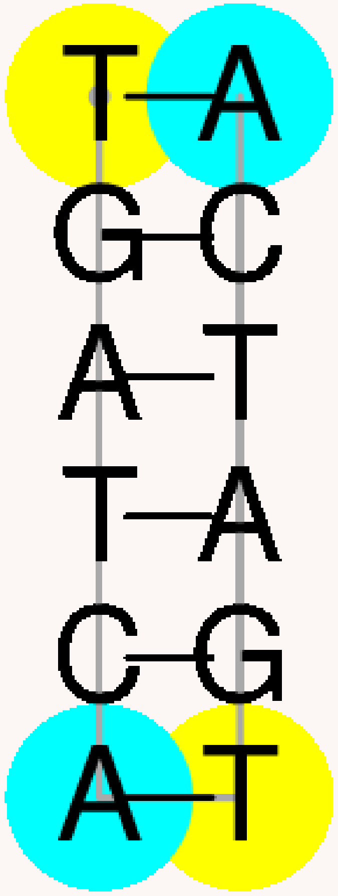


Original, MCC = 0



Mutant, MCC 0.803219

Symmetric



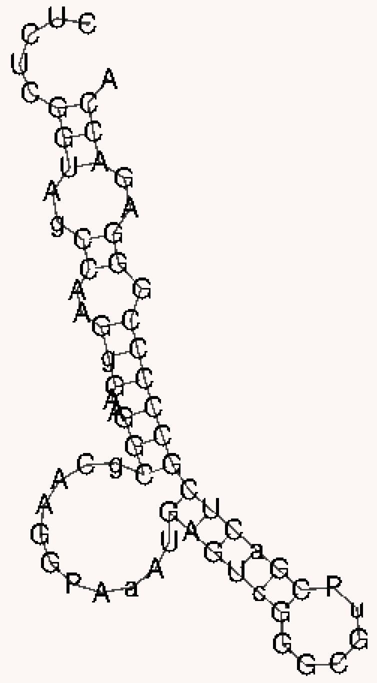
True

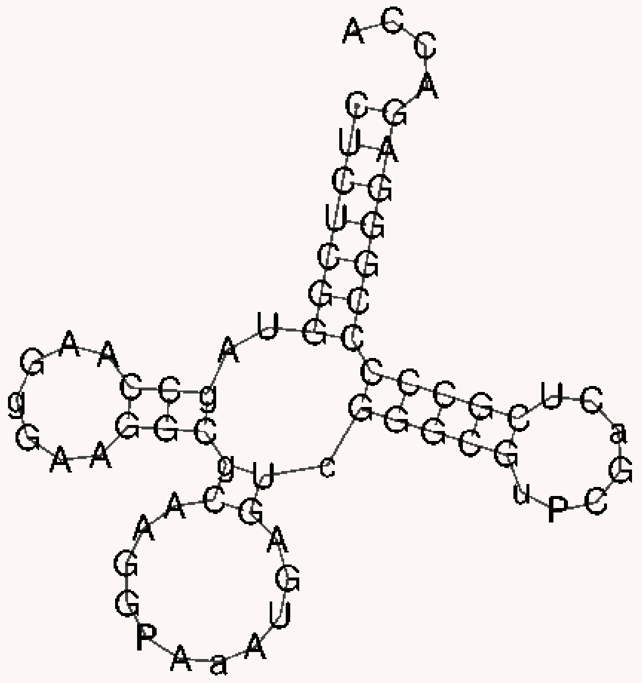
U = T

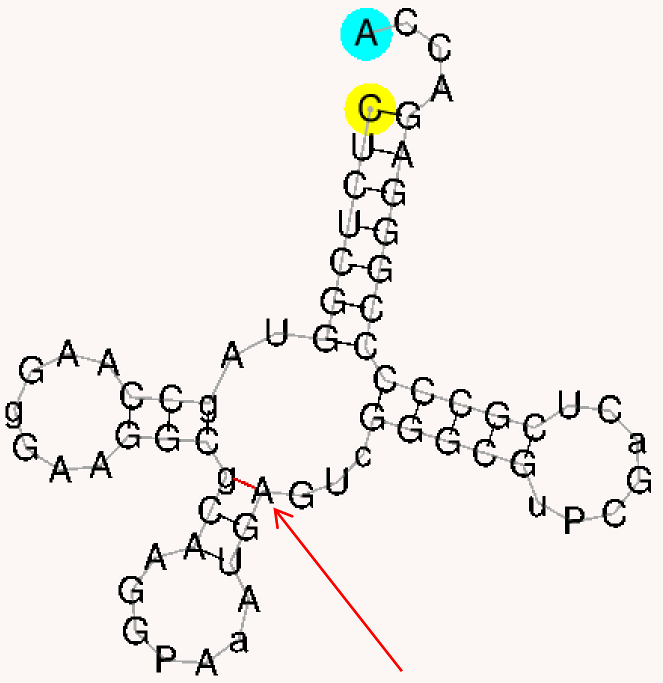
CREST

PDB_01001

yeast enzyme (in protein manufacture)







Non-standard binding

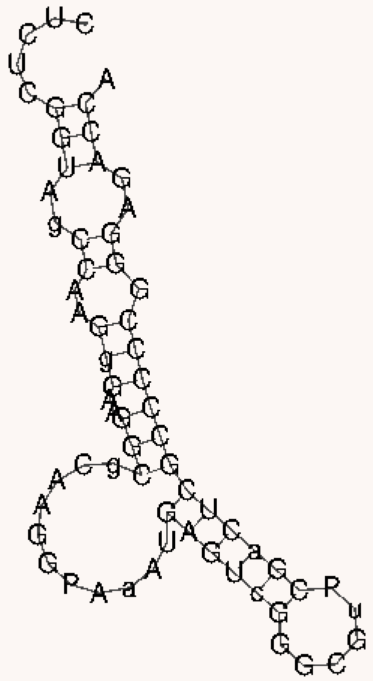
Original, MCC -0.008222

Mutant, MCC 0.856324

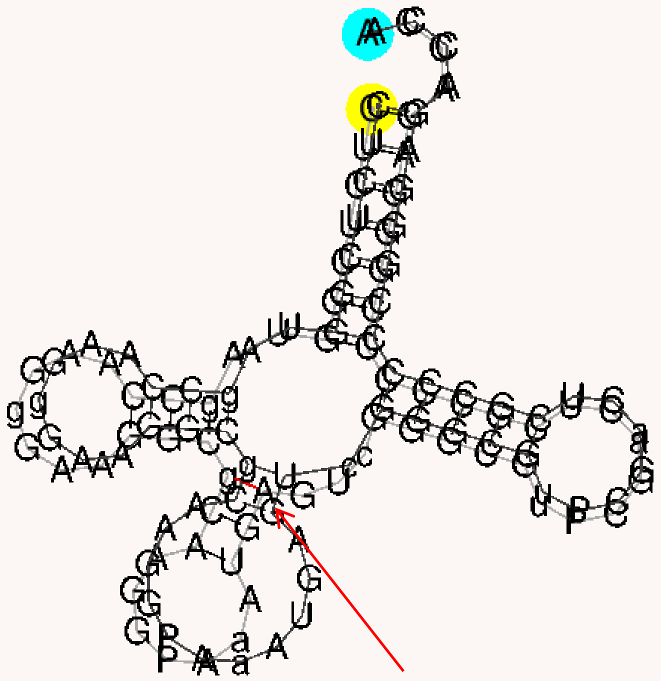
True

PDB_01001

yeast enzyme (in protein manufacture)



Original, MCC -0.008222



Mutant, MCC 0.856324

True

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 software
- GI for automated parameter update

  bugfixing, software transplanting, performance improvement, faster answers or better answers.

  BarraCUDA 3,095 sourceforge downloads (26 months).

  Commercial use by Lab7 (in BioBuilds Nov 2015)

  IBM Power8.

  RNAfold **data**,SSE,CUDA (17,061 downloads)

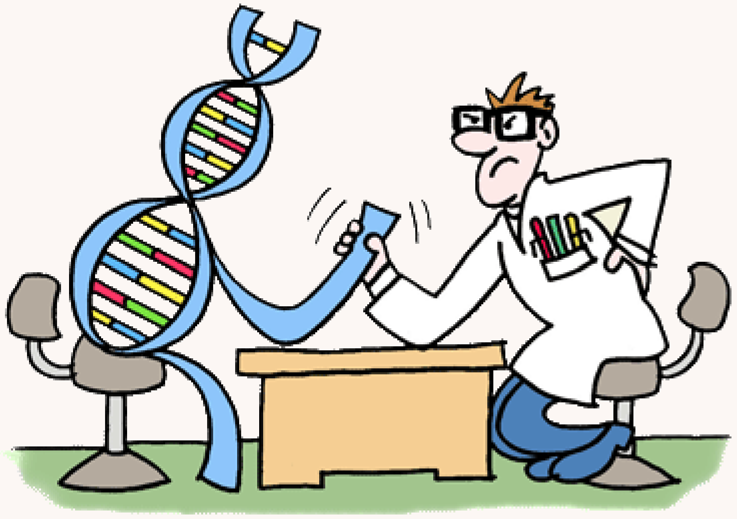
- Software is not fragile

  break it, bend it, Evolve it

CREST

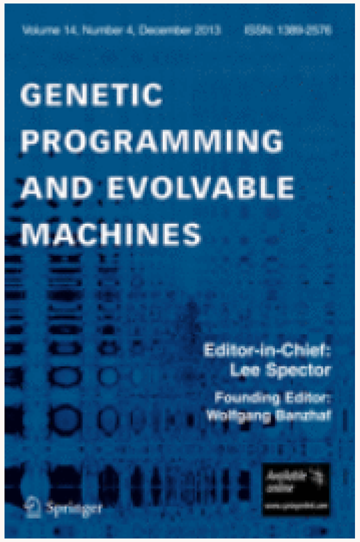
GI 2019

GI 2019, Montreal, ICSE-2019 workshop.
Submission (2 or 8 pages) due 1st February 2019



Humies: Human-Competitive
$10,000 prizes. Finals in Prague

Wikipedia
Genetic Improvement


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

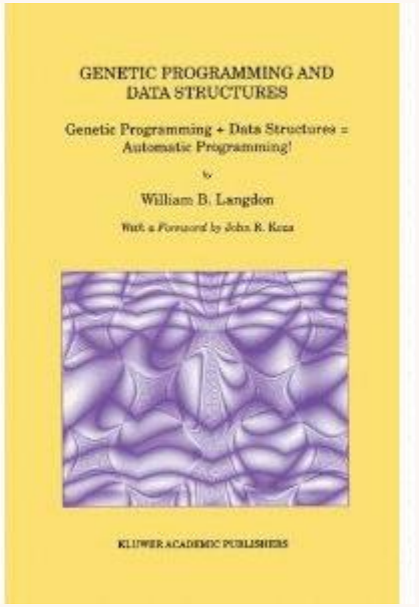
W. B. Langdon, UCL

EPSRC

W. B. Langdon

CREST

Department of Computer Science



GENETIC PROGRAMMING AND DATA STRUCTURES

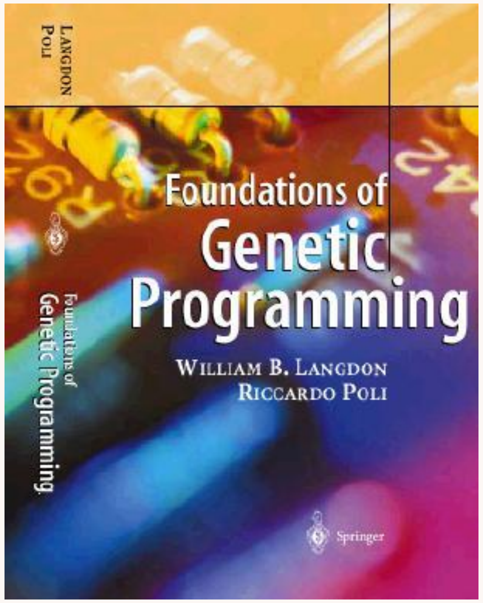
Genetic Programming + Data Structures = Automatic Programming!

by

William B. Langdon

With a Foreword by John R. Koza

KLUWER ACADEMIC PUBLISHERS

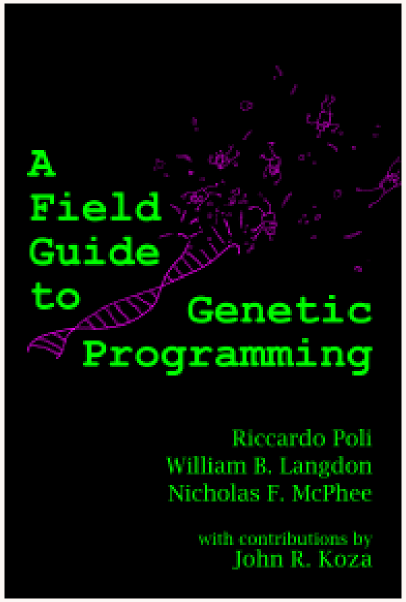


Foundations of Genetic Programming

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

Matthews Correlation Coefficient (MCC)

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

- Correlation between observed and predicted
- Range -1.00 to +1.00
- 0 no correlation
- Large class imbalance
  - Almost all possible pairs are not observed and correctly predicted as being absent
  - Number of true negatives (TN) is huge

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?

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 of RNAfold

RNAfold state of the art prediction of how RNA molecule will fold up based on its sequence of bases.

- Speed up via Intel SSE parallel instructions GI 2017. Shipped since V2.3.5
- ViennaRNA Package v2.3.0cuda
- Better predictions by evolving parameters
  - On average better predictions of RNA folding.
  - Shipped since 2.4.7
- AVX parallel speedup due release 2.4.11

The Genetic Programming Bibliography

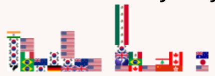
http://www.cs.bham.ac.uk/~wbl/biblio/

12667 references, 11000 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

XML RSS

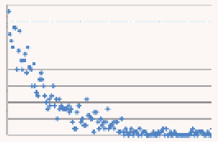
RSS Support available through the Collection of CS Bibliographies.



Downloads by day

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