CREST

Fitness Landscape of the Triangle Program

UCL Computer Science research note RN/16/05

Workshop on Landscape-Aware Heuristic Search, N.Veerapen and G.Ochoa, PPSN-2016

W. B. Langdon

Department of Computer Science

UCL

Fitness Landscape of the Triangle Program

UCL Computer Science research note RN/16/05

W. B. Langdon

Department of Computer Science

Fitness Landscape of the Triangle Program

- Background
  - Fitness landscapes of genetic improvement
  - What is the Triangle program.
  - Constructing Triangle’s fitness landscape
- Results
  - 1st order schema analysis
  - Visualisation
- Where next?
  - Benchmarks
- Insight into Genetic Improvement?

Genetic Improvement

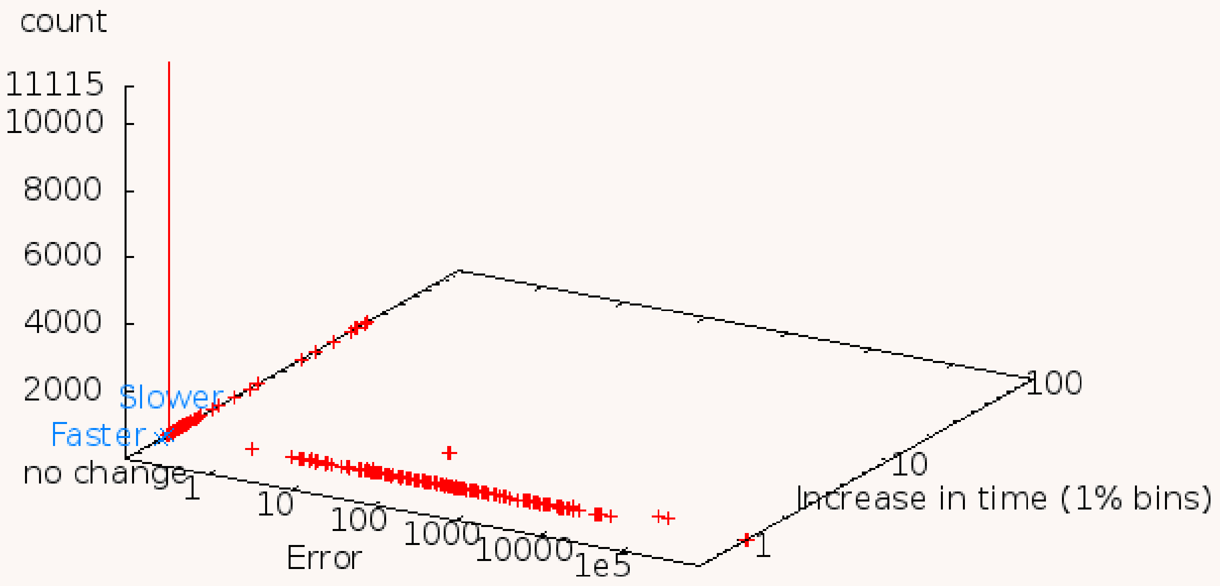
- Genetic Improvement is the application of search (often genetic programming) to improve existing software, e.g.
  - Fix bugs
  - Faster (CPU or on parallel hardware: GPU)
  - Less energy used
  - Less memory
- Real programs (10^4 to 10^6 of lines of code)

WIKIPEDIA

Genetic Improvement

GI Mutation Fitness Landscapes

14,173 Successful single code mutations to BWA on execution path



89% mutations which compile make no change to test case CS-DC'15

BWA 0.7.12-r1039
10958 lines of code

GI Fitness Landscapes

- Real software is resilient to mutations.
- Schema (crossover) analysis.
- The Triangle program is a software engineering benchmark

Triangle Program

- Given length of three sides what type is triangle? (Software Engineering benchmark)

- Test suite covers all paths [JSS 83(12) (2010) 2416–2430]
- Mutate conditionals
- Fitness is number of tests that still pass
- Simplified so can *enumerate all* mutations
- UCL-CS [RN/16/05] dataset online http://www.cs.ucl.ac.uk/staff/W.Langdon/ppsn2016/triangle/
- Real program, whole landscape, try your tool

```
int gettri(int side1, int side2, int side3)
{

     int triang ;

    if( side1 <= 0 || side2 <= 0 || side3 <= 0){
        return 4;
    }

    triang = 0;

    if(side1 == side2){
        triang = triang + 1;
    }
    if(side1 == side3){
        triang = triang + 2;
    }
    if(side2 == side3){
        triang = triang + 3;
    }

    if(triang == 0){
        if(side1 + side2 <= side3 ||
 side2 + side3 <= side1 || side1 + side3 <= side2){
            return 4;
        }
        else {
            return 1;
        }

    }

    if(triang > 3){
        return 3;
    }
    else if ( triang == 1 && side1 + side2 > side3) {
        return 2;
    }
    else if (triang == 2 && side1 + side3 > side2){
        return 2;
    }
    else if (triang == 3 && side2 + side3 > side1){
        return 2;
    }

    return 4;
}
```

All comparisons are potential mutation sites. Shown in red

== replaced by <=
 > replaced by !=
<= replaced by ==

(Chosen as they are the hardest to detect mutations.)

```
int gettri(int side1, int side2, int side3)
{

     int triang ;

    if( side1 <= 0 || side2 <= 0 || side3 <= 0){
        return 4;
    }

    triang = 0;

    if(side1 == side2){
        triang = triang + 1;
    }
    if(side1 == side3){
        triang = triang + 2;
    }
    if(side2 == side3){
        triang = triang + 3;
    }

    if(triang == 0){
        if(side1 + side2 <= side3 ||
 side2 + side3 <= side1 || side1 + side3 <= side2){
            return 4;
        }
        else {
            return 1;
        }

    }

    if(triang > 3){
        return 3;
    }
    else if ( triang == 1 && side1 + side2 > side3) {
        return 2;
    }
    else if (triang == 2 && side1 + side3 > side2){
        return 2;
    }
    else if (triang == 3 && side2 + side3 > side1){
        return 2;
    }

    return 4;
}
```

testcases_oracle.txt

Inputs are the three sides of the triangle.

Output is correct classification of the triangle.

Test suite covers all paths but is not strong enough to detect all mutations.

Dataset gives whole test equivalent fitness landscape

Three inputs	expected output
0 0 0	4
1 0 0	4
1 1 0	4
1 1 1	3
2 2 1	2
1 1 2	4
2 1 2	2
1 2 1	4
2 1 1	4
3 2 2	2
3 2 1	4
4 3 2	1
2 3 1	4
2 1 3	4

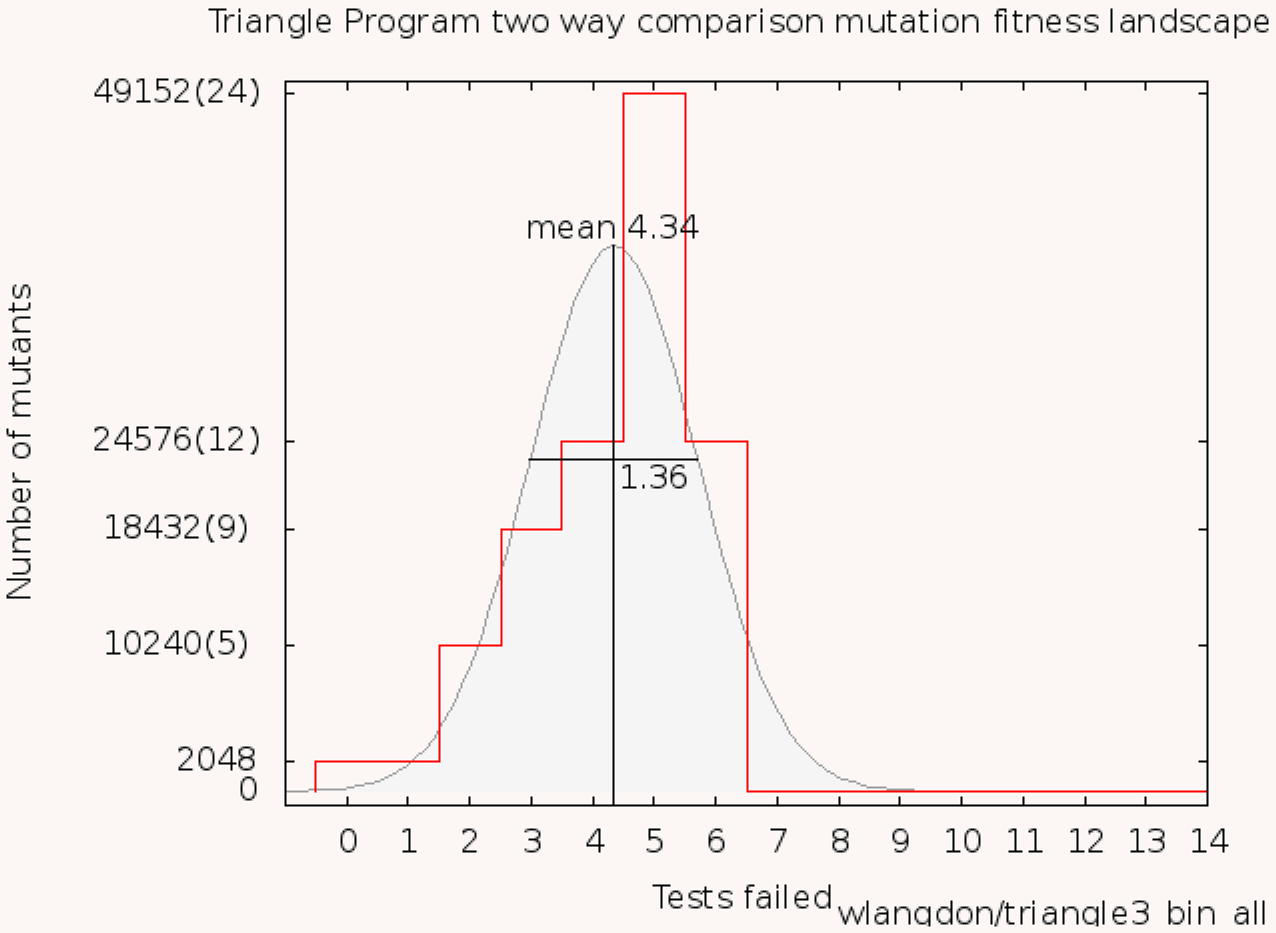
Triangle Fitness Landscape

- Only mutate C comparison operators (17)
- Reduce space from 6^{17} to 2^{17} by allowing only hardest to detect mutation. (I.e. one change rather than five.)
- Run all tests (14). For how many does new code give the wrong answer? (0-6)

Triangle Program search space

- On average (zeroth order schema) each mutant fails only 4.34 tests (standard deviation 1.36)
- 2048 global optima
- Worst mutant fails 6 of the 14 tests

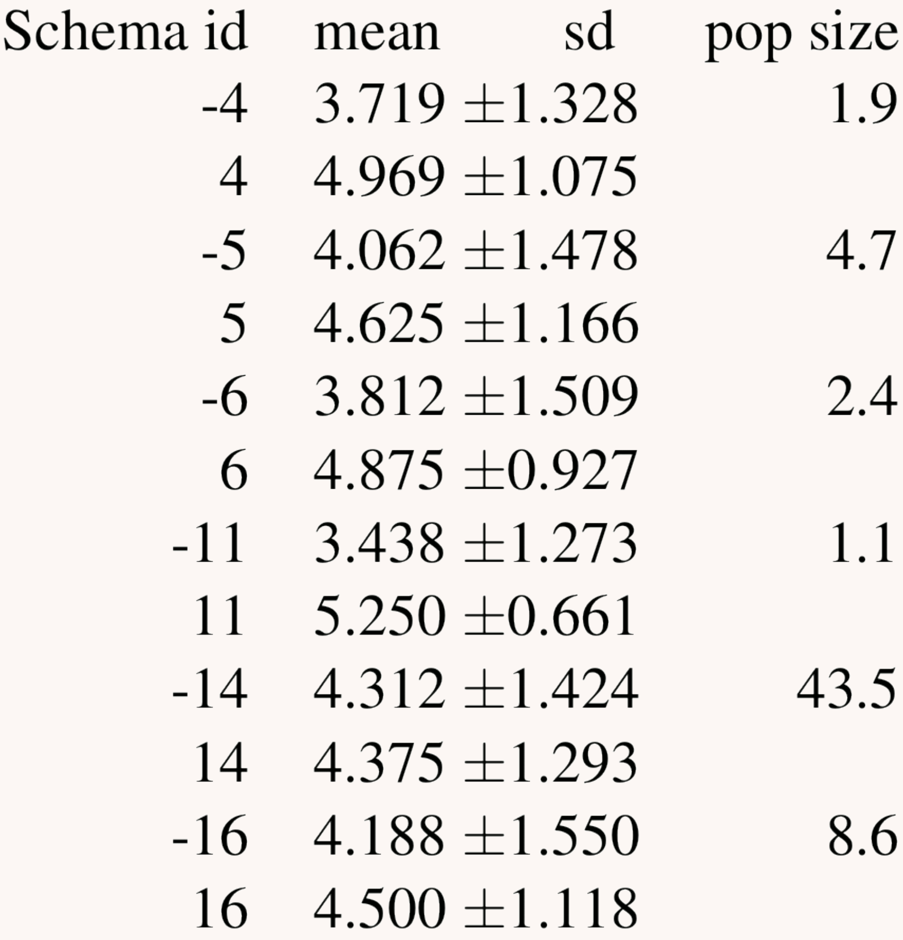
How many mutants failed how many tests



First Order Schema

- 34 1st order schema
- 22 have exactly average fitness and contain exactly half the global optima.
  I.e. no useful signal.
- 6 schema worse than average, no optima
- 6 better than average, each contains all optima
- 1st order schema are not deceptive.

First Order Schema

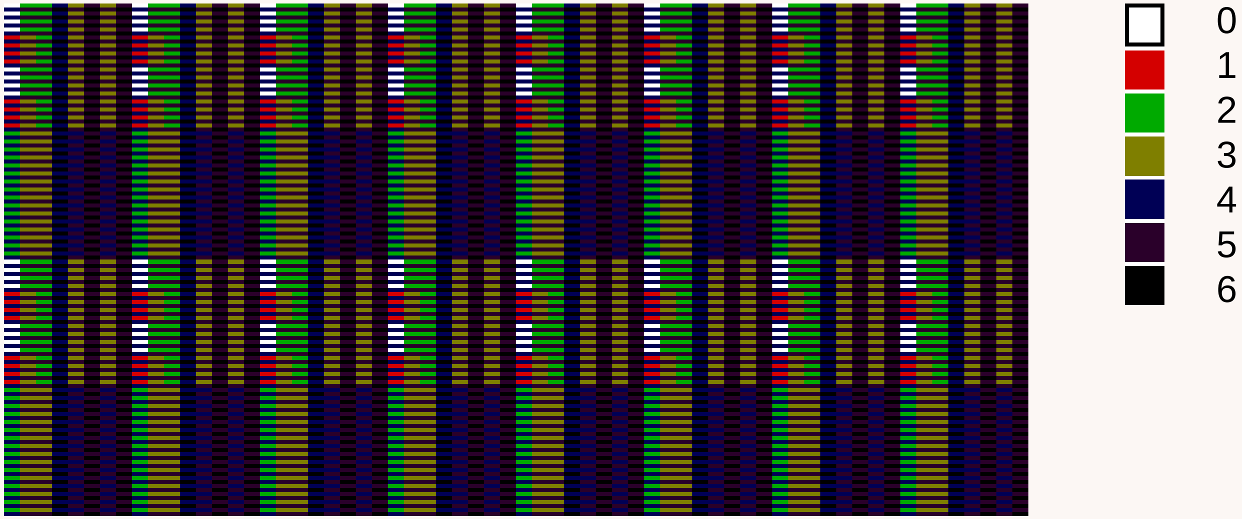


Schema id	mean	sd	pop size
-4	3.719	±1.328	1.9
4	4.969	±1.075	
-5	4.062	±1.478	4.7
5	4.625	±1.166	
-6	3.812	±1.509	2.4
6	4.875	±0.927	
-11	3.438	±1.273	1.1
11	5.250	±0.661	
-14	4.312	±1.424	43.5
14	4.375	±1.293	
-16	4.188	±1.550	8.6
16	4.500	±1.118	

Pop size = $2sd_{12} / |mean_1 - mean_2|$

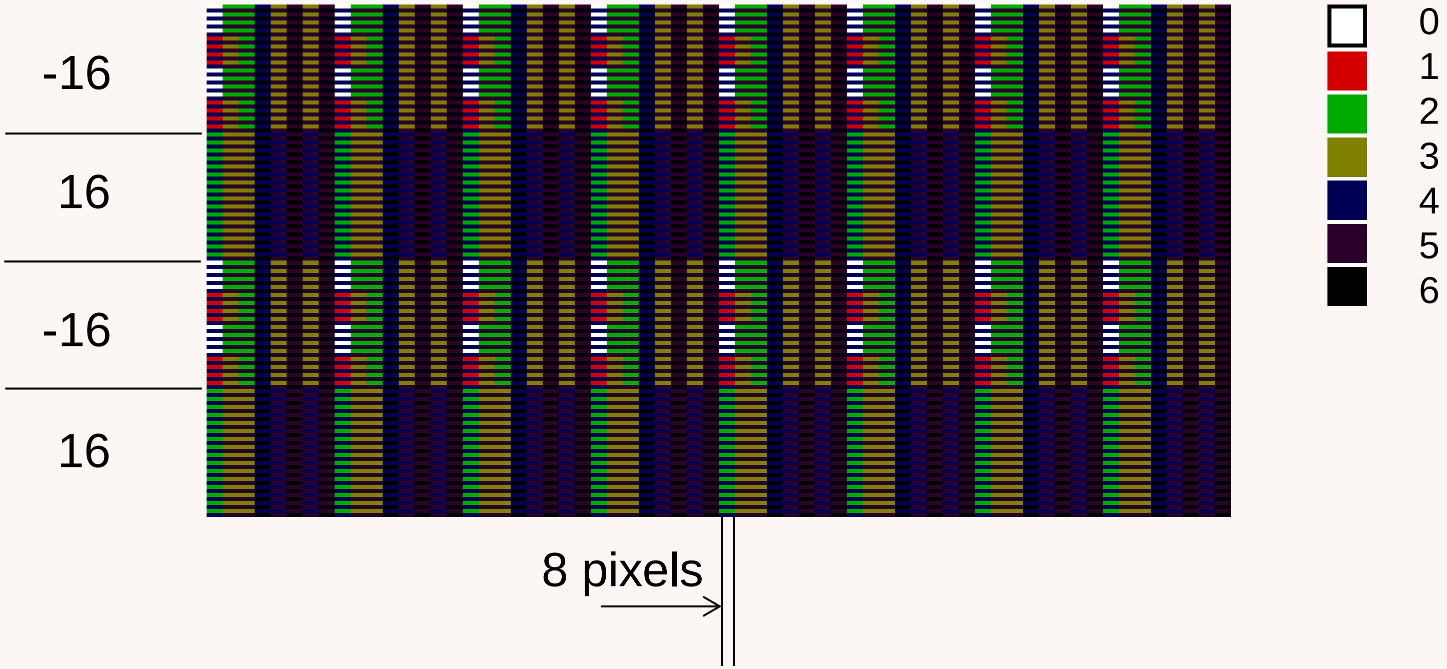
Table 2: Mean and standard deviation of number of tests failed for first order schema (excluding 22 with average means).

Fitness in bit order ($2^8 \times 2^9$)



- 2048 global optima in white.
- Regular patterns indicate small building blocks.
- Vertical strips 8 pixels wide says first three bits do not impact fitness.
- Last but one bit gives four horizontal stripes:
  - two contain 50176 mutants fail ≥4 tests (dark)
  - others hold all the solutions (white)

Fitness in bit order ($2^8 \times 2^9$)



- Vertical strips 8 pixels wide says first three bits do not impact fitness.
- Last but one bit gives four horizontal stripes:
  - two contain 50176 mutants fail ≥4 tests (dark)
  - others hold all the solutions (white)

Genetic Improvement Benchmarks

- Bugs to be fixed
  - GenProg http://dijkstra.cs.virginia.edu/genprog/
- Software Engineering

  Many, e.g. SIR http://sir.unl.edu
- Fitness landscape

  Mutation testing/GA fitness landscape for the Triangle Program, UCL CS RN/16/05

  http://www.cs.ucl.ac.uk/staff/W.Langdon/ppsn2016/triangle/

Insight into Genetic Improvement?

- Success of genetic improvement may be due to programs not being as fragile as is often assumed.
- Mutation only studies C++ code is robust
- GI search >> random but not yet best?
- 1st GI schema analysis. Triangle program real software engineering benchmark
  - not deceptive
  - fitness landscape data available

END

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

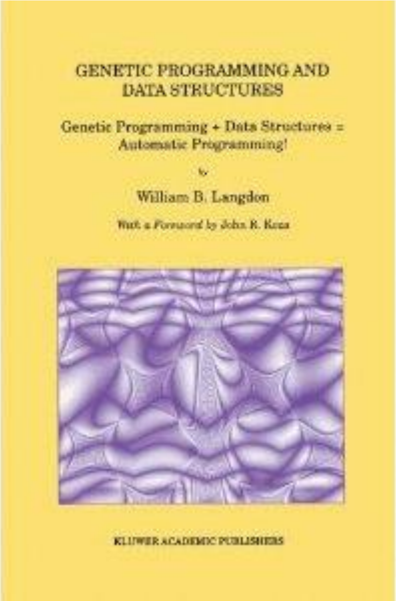
Genetic Improvement

UCL

W. B. Langdon

CREST

Department of Computer Science



GENETIC PROGRAMMING AND DATA STRUCTURES

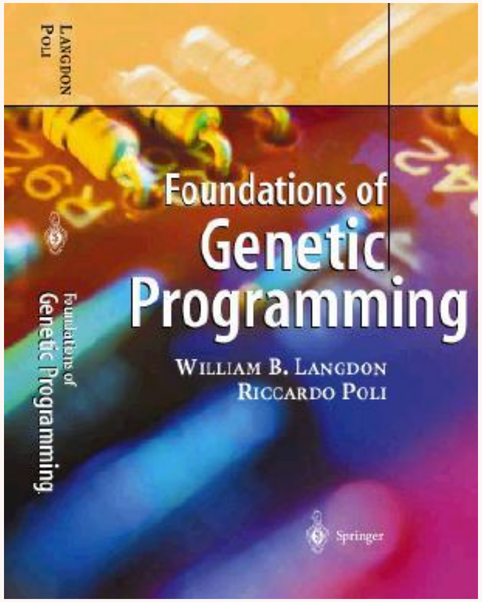
Genetic Programming + Data Structures = Automatic Programming!

by

William B. Langdon

With a Foreword by John R. Koza

KLUWER ACADEMIC PUBLISHERS



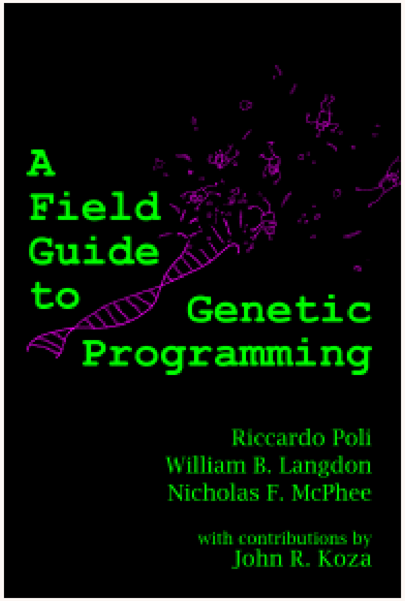
LANGDON POLI

Foundations of Genetic Programming

WILLIAM B. LANGDON
RICCARDO POLI

Springer

Foundations of Genetic Programming

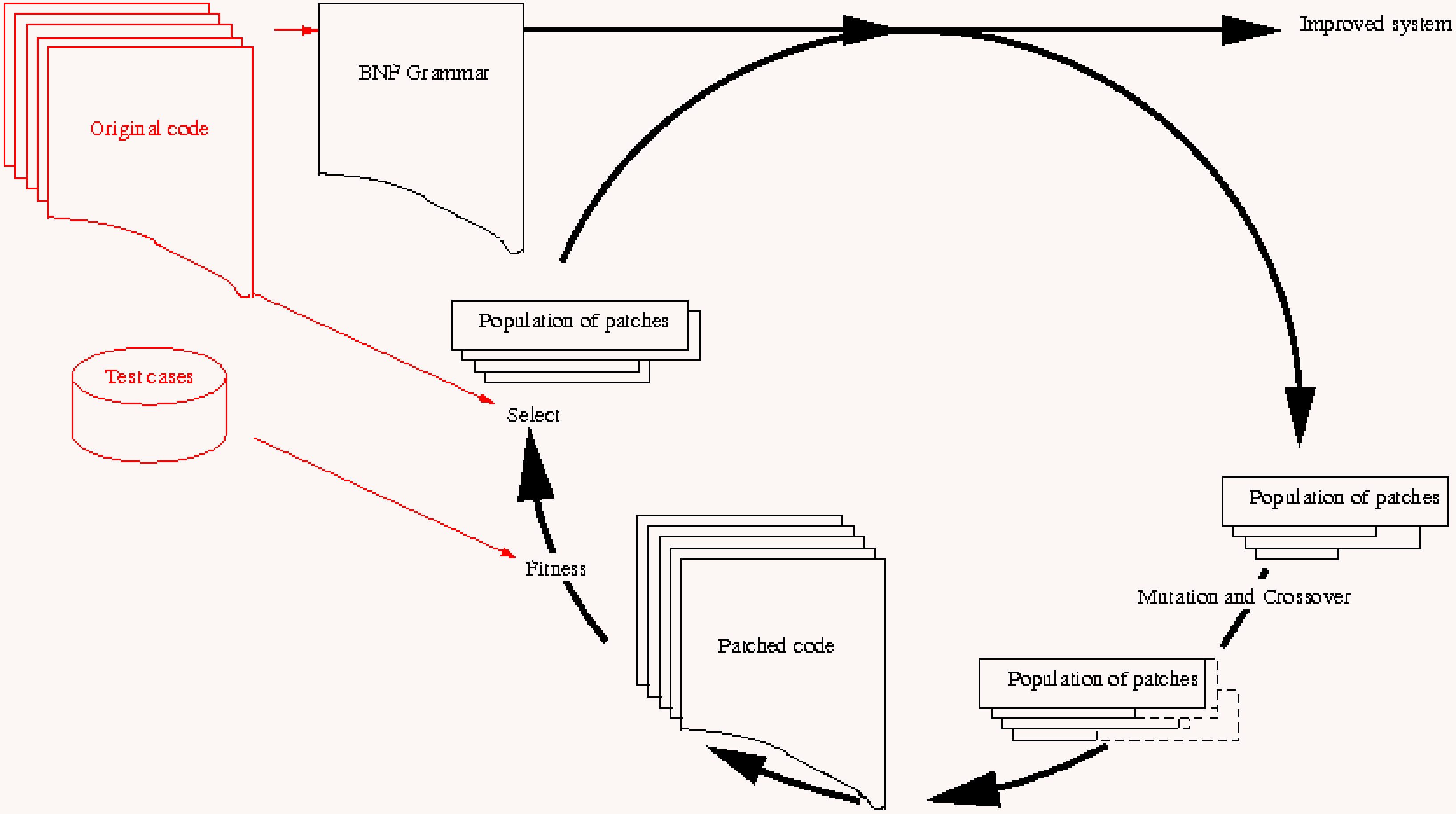


A Field Guide to Genetic Programming

Riccardo Poli
William B. Langdon
Nicholas F. McPhee

with contributions by
John R. Koza

Genetic Improvement evolves code patches



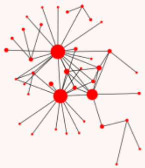
Recent Successes of Genetic Improvement

- Automatic bug repair
  - GenProg, e.g.105 bugs fix most (multiple best papers, IFIP TC2 Manfred Paul Award, 2 Humies)
- Better programs
  - 70x Bowtie2, BarraCUDA, pknots 10000x
  - Less energy, less memory
  - MOGA speed v. quality, e.g. [SIGGRAPH]
- Code transplant [Marginean, e.g. best paper ISSTA 2015]
  - E.g. C++, code indent, call graph layout into Kate editor (we *can* evolve an editor) Humie

The Genetic Programming Bibliography

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

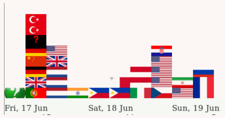
11246 references



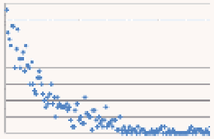
Part of gp-bibliography 84 40 Revision:1.1794 29 May 2011

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.



blog

Google scholar citations

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