



Genetic improvement: Taking real-world source code and improving it using genetic programming

Sæmundur Ó. Haraldsson,

UNIVERSITY of STIRLING

John R. Woodward,

Operational Research
Queen Mary University of London

Markus Wagner

THE UNIVERSITY of ADELAIDE

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.

GECCO '20 Companion, July 8–12, 2020, Cancún, Mexico

© 2020 Copyright is held by the owner/author(s).

ACM ISBN 978-1-4503-7127-8/20/07.

<https://doi.org/10.1145/3377929.3389885>



DE160100850

GECCO 2020

Overview

● Introduction

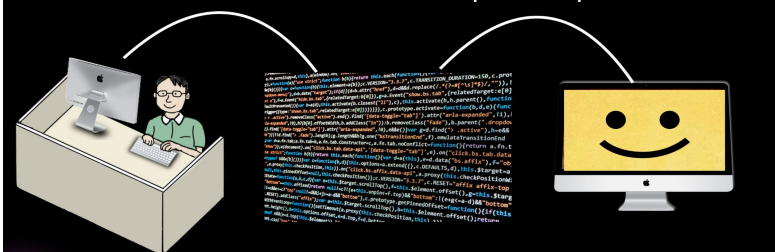
- Fixing Bugs and other examples
- Noteworthy papers and issues
- Getting involved
- Summary and Q&A

2

Genetic Improvement of Software

human writes code

computer improves it

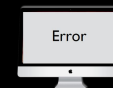


Functional Properties

LOGICAL



New Feature



Bug Repair

accuracy

Non-Functional Properties

PHYSICAL



Execution Time



Memory



Bandwidth



Battery



Size

There is nothing correct about a flat battery (BILL LANGDON)

What is Genetic Improvement

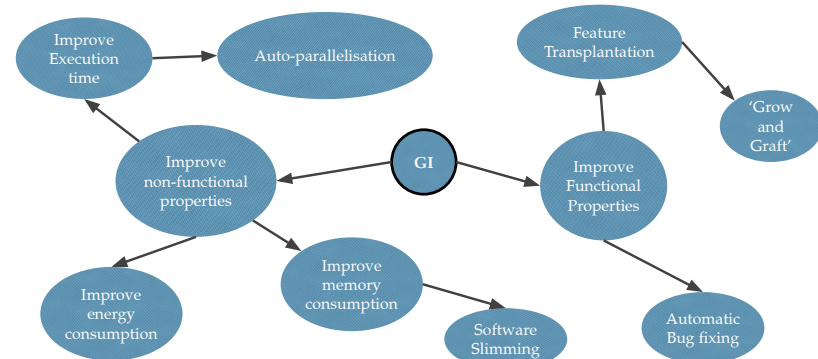


A wordy definition:

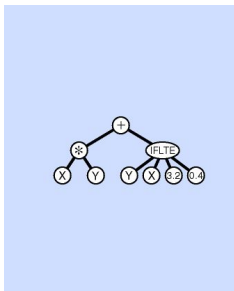
Genetic Improvement is the application of search-based (typically evolutionary) techniques to modify software with respect to some user-defined fitness measure.

It's just GP - BUT starting with a nearly complete program
[Wolfgang Banzhaf]

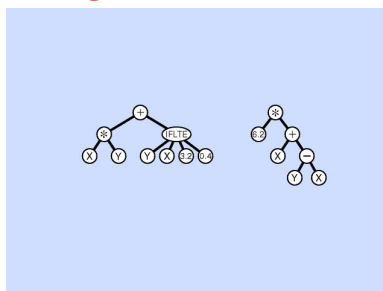
What is Genetic Improvement



Genetic Programming overview



mutation



crossover

Genetic Programming: GI's ROOTS

1. **Aim** – to discover new programs by telling the computer what we want it to do, but not how we want it to do it – John Koza
2. **How** – we evolve computer programs using natural selection.
3. **Starts from scratch** (empty program)
4. Choose **primitives** (terminal set/FEATURES and function set)
5. Choose **representation** (tree based, graph based, linear e.g. CGP)
6. Choose **fitness function**, parameters, genetic operators.

GI forces “the full capabilities of programming languages”- side effects, ADFs, LOOPS

GP vs GI: if you can't beat them, join them.

John R. Woodward
University of Stirling
Stirling
Scotland, United Kingdom
jrw@cs.stir.ac.uk

Colin G. Johnson
University of Kent
Kent
England, United Kingdom
C.G.Johnson@kent.ac.uk

Alexander E.I. Brownlee
University of Stirling
Stirling
Scotland, United Kingdom
sbr@cs.stir.ac.uk

ABSTRACT

Genetic Programming (GP) has been criticized for targeting irrelevant problems [12], and is true of the wider machine

(procedures, methods, macros, routines), and so GI has to deal with the reality of existing software systems. However, most of the GP literature is not concerned with Tur-

Popular Science

- easy to digest articles for non-specialists.

<https://theconversation.com/computers-will-soon-be-able-to-fix-themselves-are-it-departments-for-the-chop-85632>

Computers will soon be able to fix themselves – are IT departments for the chop?

October 12, 2017 3:29pm BST

IT?

Authors



Saemundur Haraldsson
Postdoctoral Research Fellow,
University of Stirling



Alexander Brownlee
Senior Research Assistant,
University of Stirling



John R. Woodward
Lecturer in Computer Science,
Queen Mary University of London



<https://theconversation.com/how-computers-are-learning-to-make-human-software-work-more-efficiently-43798>

How computers are learning to make human software work more efficiently

June 25, 2015 10:08am BST

Authors



John R. Woodward
Lecturer in Computer Science,
University of Stirling



Justyna Petke
Research Associate at the Centre
for Research on Evolution, Search
and Testing, UCL



William Langdon
Principal Research Associate,
UCL



<http://www.davidrwhite.co.uk/2014/11/27/genetic-programming-has-gone-backwards/>



Genetic Programming has gone Backwards

When Genetic Programming (GP) first arose in the late 80s and early 90s, there was one very defining characteristic of its application, which was so widely accepted as to be left unsaid:

GP always starts from scratch

<http://www.davidrwhite.co.uk/tag/genetic-programming/>



This blog post is based on a seminar given to the Department of Computer Science at the University of Manchester in April 2016; it also builds on the ideas and talks of many fellow academics, who I acknowledge at the end of the article.

THE CONVERSATION

Academic rigour, journalistic flair

Search analysis, research, academics...

Arts + Culture Business + Economy Cities Education Environment + Energy Health + Medicine Politics + Society **Science + Technology** Brexit

Never mind the iPhone X, battery life could soon take a great leap forward

September 13, 2017 2:29pm BST



Authors



Alexander Brownlee
Senior Research Assistant,
University of Stirling



Jerry Swan

Competent Programmers Hypothesis

1. programmers write programs that are almost perfect.
2. program faults are syntactically small (slip of finger, T/F)
3. corrected with a few keystrokes. (e.g. < for <=)
4. **GI can find small patches.**
5. Small changes are non-unique (7 lines code, or utter 7 words **before** **unique**)

Plastic Surgery Hypothesis.

the content of new code can often be assembled out of fragments of code that already exist.

Barr et al. [71] showed that changes are 43% graftable from the exact version of the software being changed.

The Plastic Surgery Hypothesis: Changes to a codebase contain snippets that already exist in the codebase at the time of the change, and these snippets can be efficiently found and exploited.

THE CODE CONTAINS SOLUTIONS – CANDIDATE PATCHES

Representations of PROGRAMS

Natural Representation of CODE

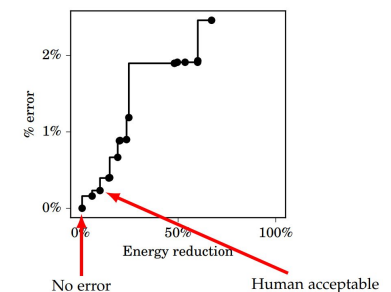
1. Text files e.g. Program.java is a text file. Saemi.
2. Abstract syntax tree (AST) – Genprog, Genofix.
3. Java byte code (also C binaries) [102]
4. Errors, compile, halting (Langdon - discard)

Objectives

- Functional (**logical properties**)
 - Accuracy e.g. as in machine learning - FLOAT
 - Number of bugs – as measured against a set of test cases. BOOLEAN
 - New functionality – e.g.
- Non-functional (**physical properties**)
 - Execution time
 - Energy (power consumption – peak/average)
 - Memory
 - Bandwidth
- Multi-objective
 - Trade-offs, convex, a set of programs = a single tuneable program

Multi-Objective

- Seems be convex
- – simple argument (see pic)
- Can provide a set of programs
- weighted sum of objectives?
- weight have meaning to user.
- Will there be elbow/knee points?



Slow connections.



Loading Gmail

Loading standard view | [Load basic HTML](#) (for slow connections)



GISMOE

The GISMOE challenge:

to create an automated program development environment in which the Pareto program surface is automatically constructed to support dialog with and decision making by the software designer concerning the trade offs present in the solution space of programs for a specific programming problem.

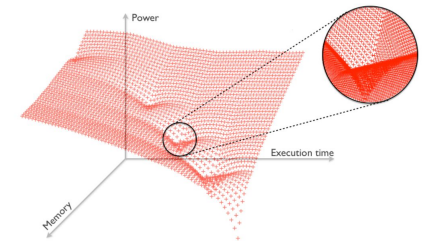


Figure 1: The GISMOE Pareto Program Surface

EDITS Operators – changes to programs

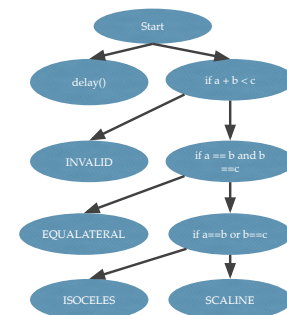
- Line level
- Single Character level
- Function/module level.
- AST – GIN, Gen-0-fix, genprog,
- Java – machine code – java byte code.
- LIST OF EDITS IS A PATCH.

GI: An example of execution time optimisation

```
static final int INVALID = 0;
static final int SCALENE = 1;
static final int EQUALATERAL = 2;
static final int ISOCELES = 3;

public static int classifyTriangle(int a, int b, int c) {
    delay();
    assert(a <= b && b <= c);
    if (a + b <= c) {
        return INVALID;
    } else if (a == b && b == c) {
        return EQUALATERAL;
    } else if (a == b || b == c) {
        return ISOCELES;
    } else {
        return SCALENE;
    }
}

private static void delay() {
    try {
        Thread.sleep(100);
    } catch (InterruptedException e) {
        // do nothing
    }
}
```

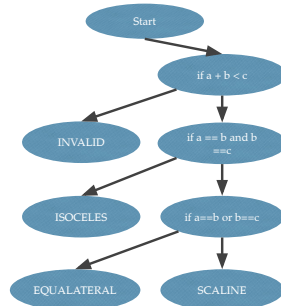


GI: An example of automated bug fixing

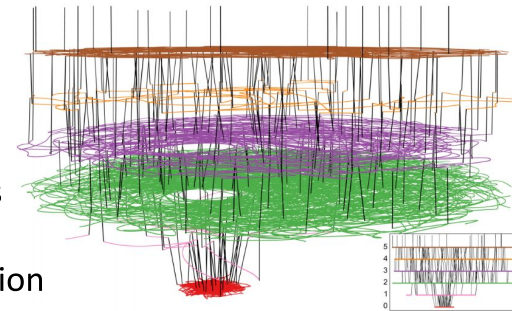
```
static final int INVALID = 0;
static final int SCALENE = 1;
static final int EQUALATERAL = 2;
static final int ISOCELES = 3;

public static int classifyTriangle(int a, int b, int c) {
    assert(a <= b && b <= c);
    if (a + b <= c) {
        return INVALID;
    } else if (a == b && b == c) {
        return ISOCELES;
    } else if (a == b || b == c) {
        return EQUALATERAL;
    } else {
        return SCALENE;
    }
}

private static void delay() {
    try {
        Thread.sleep(100);
    } catch (InterruptedException e) {
        // do nothing
    }
}
```



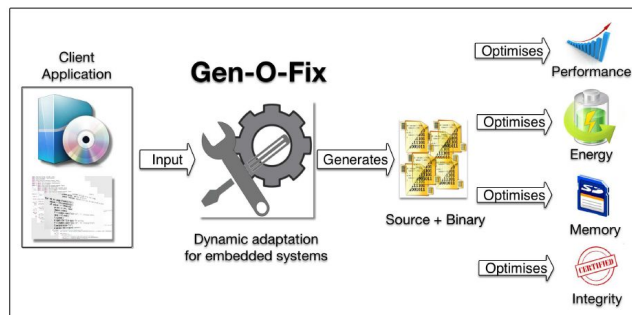
Neural
networks
Graceful
degradation



structure
Hill
climber

Fig. 1. Local optima network of the Triangle Program using 100 random starts (see Section 4.4). Edges are coloured if they start and end at the same fitness. Insert shows fitness levels edge on. Best (bottom) red 0 (pass all tests), pink 1 (fail only one test), green 2, purple 3, orange 4, brown 5.

System Diagram for Gen-O-Fix



Gen-O-Fix: Abstract Syntax Trees

Main features of framework are

1. **Embedded** adaptively.
 1. Initial source code: **location** of Scala source code file containing a function
 2. Fitness function: providing a means of **evaluating the quality** of system
3. **Source to source transformations**
4. Operates on ASTs (i.e. arbitrarily fine).

AST - scala

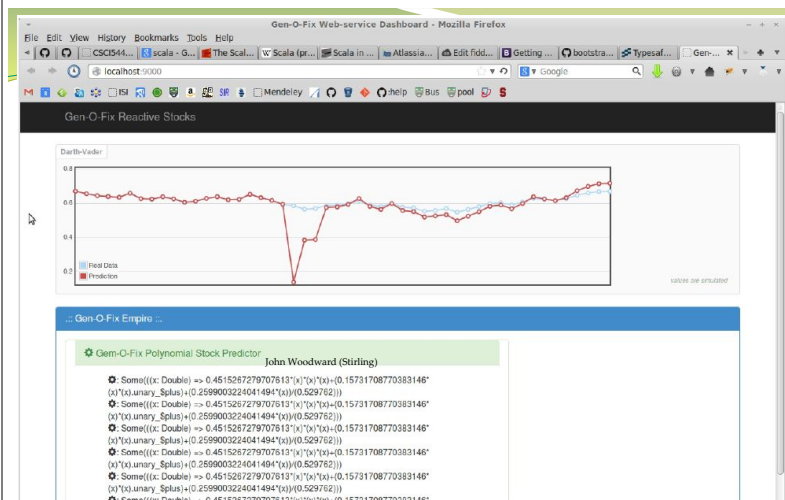
Code as data, data as code.

```
// code to data:
var m = 2; var x = 3; var c = 4
val expr = reify( ( m * x ) + c )
println( "AST = " + showRaw( expr.tree ) )

// output:
AST = Apply( Select( Apply( Select( Select( Ident("m"),
"elem"), "$times"), List( Select( Ident("x"),
"elem") ) ), "$plus"), List( Select( Ident("c"),
"elem") ) ) )
```

```
// run AST datatype as code:
println( "eval = " + expr.tree.eval() )

// output:
eval = 10
```



GI Hashcode

1. **Hadoop** provides a mapReduce implementation in Java.
2. Equals method has to obey **contract** (Reflective, Symmetric, Transitive, ...)
3. `x.equals(y)` **implies** `hashCode(x) == hashCode(y)`.
4. `hashCode` method is an integer function of a subset of an object's fields

Some GP Settings

1. **Terminal set** is
 1. Field values
 2. Random integers [0, 100]
2. **Function set** is
 1. {+, *, XOR, AND}
3. **Fitness function:** close to uniform distribution (uniform distribution is the ideal), over 10,000 instances.

Distribution of Hashcodes

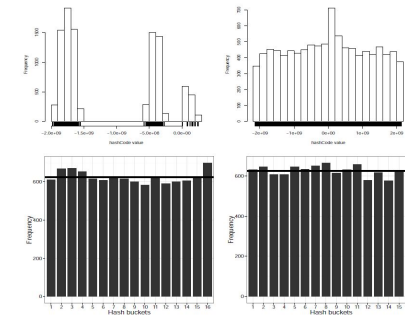


Fig. 1: The distribution of the hashcode values (top) and the distribution of the created objects in hash buckets (bottom), generated by the Apache commons (left) and the evolved function (right)

Overview

- Introduction
- **Fixing Bugs and other examples**
- Noteworthy papers and issues
- Getting involved
- Summary and Q&A

Fixing Bugs and other examples

Saemundur O. Haraldsson

- Fixing bugs
- Making software faster
- Making software more accurate



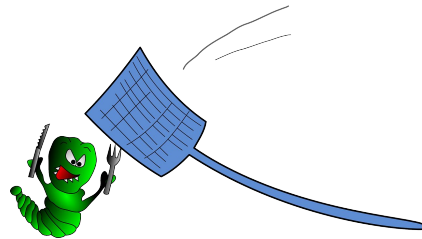
Fixing bugs

A real world example of GI in action

Saemundur O. Haraldsson, John R. Woodward, Alexander E. I. Brownlee, and Kristin Siggeirsdottir. 2017. Fixing bugs in your sleep: how genetic improvement became an overnight success. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '17). ACM, New York, NY, USA, 1513-1520. DOI: <https://doi.org/10.1145/3067695.3082517>

S.O. Haraldsson, J. R. Woodward and A. I. E. Brownlee, "The Use of Automatic Test Data Generation for Genetic Improvement in a Live System," 2017 IEEE/ACM 10th International Workshop on Search-Based Software Testing (SBST), Buenos Aires, 2017, pp. 28-31. DOI: <https://doi.org/10.1109/SBST.2017.10>

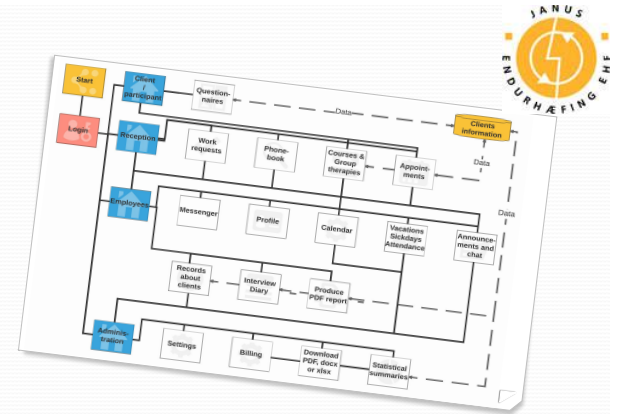
S.O. Haraldsson. 2017. 'Genetic Improvement of Software: From Program Landscapes to the Automatic Improvement of a Live System', PhD thesis, University of Stirling, Stirling. <http://hdl.handle.net/1893/26907>



37

Janus Manager

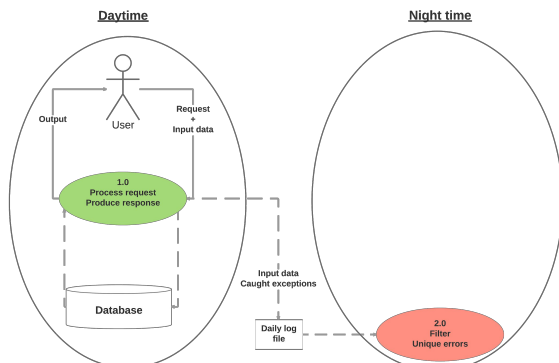
- Management system for rehabilitation
 - Python source code
 - >25K LOC
- ~200 users
 - ~40 specialists
 - 150-160 patients
- In use since March 2016
- 47 bugs automatically fixed to date



38

When last user logs out

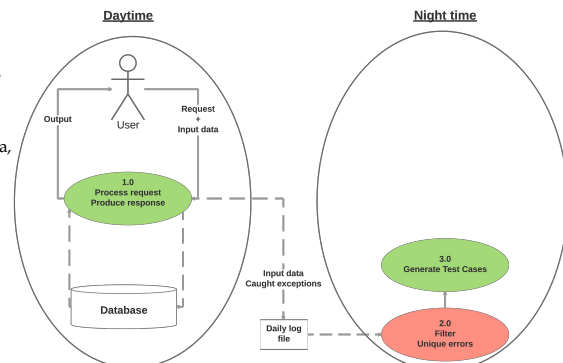
1. **Procedure 2.0**
 - Sorts and filters the day's exceptions



39

When last user logs out

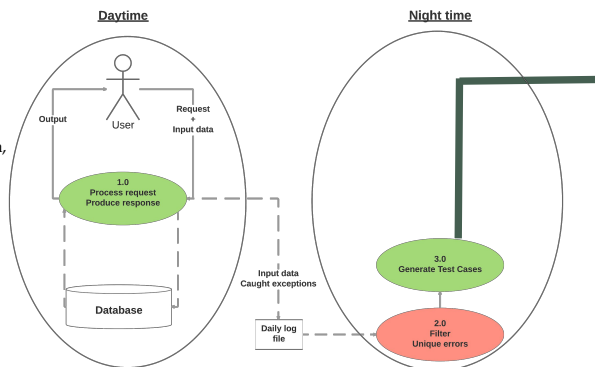
1. Procedure 2.0 started
 - Sorts and filters the day's exceptions
2. **Procedure 3.0**
 - Emulates input data, type, size and structure.
 - Produces test cases



40

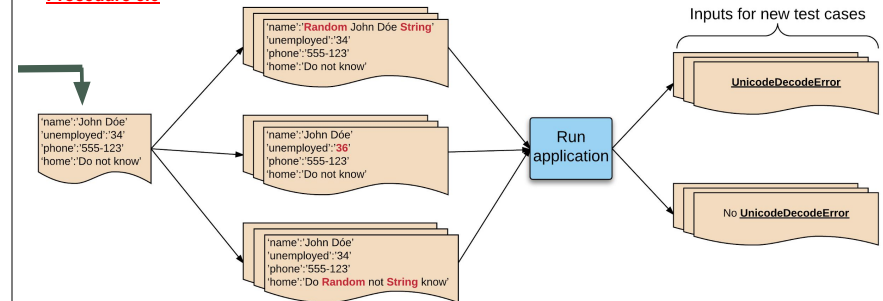
When last user logs out

1. Procedure 2.0 started
 - Sorts and filters the day's exceptions
2. **Procedure 3.0**
 - Emulates input data, type, size and structure.
 - Produces test cases



41

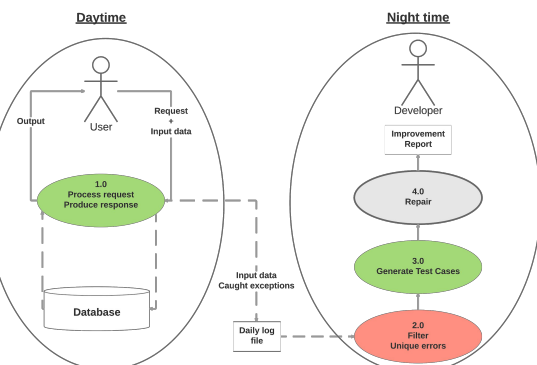
Procedure 3.0



42

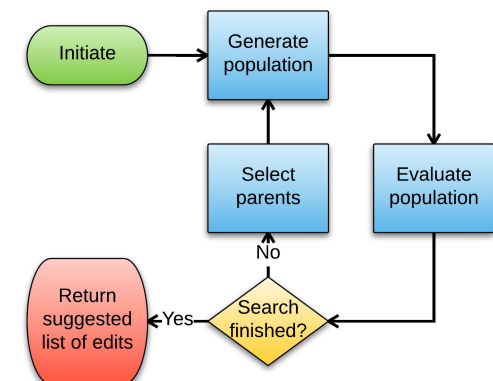
When last user logs out

1. Procedure 2.0 started
 - Sorts and filters the day's exceptions
2. Procedure 3.0
 - Emulates input data, type, size and structure.
 - Produces test cases
3. **Procedure 4.0**
 - Genetic Improvement
 - Parallel process on the server
 - Outputs report for developer



43

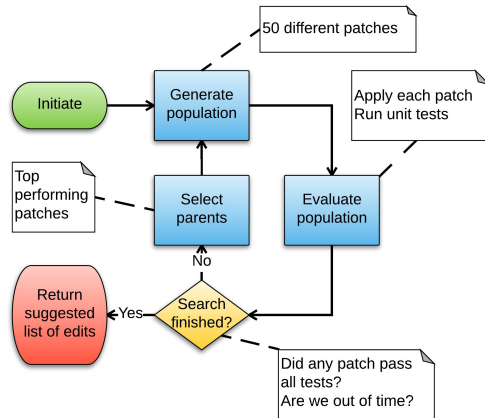
- **Procedure 4.0**
- Genetic Improvement
 - Pop.= 50 patches
 - fit.= #passed tests
 - select= 1/2 pop by fitness
 - Output= report



44

• **Procedure 4.0**

- Genetic Improvement
 - Pop.= 50 patches
 - fit.= #passed tests
 - select= 1/2 pop by fitness
 - Output= report



45

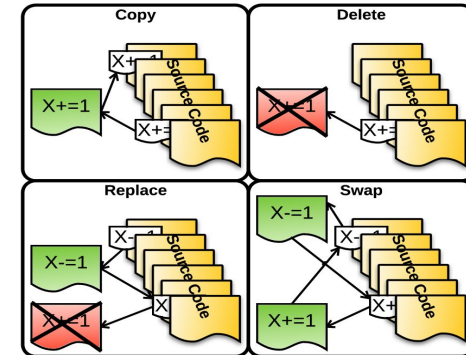
4 different types of implemented Edits

Primitive types:

- Copy**
 - Equivalent to: CTRL+C -> CTRL+V
- Delete**
 - Almost what you think

Composite types:

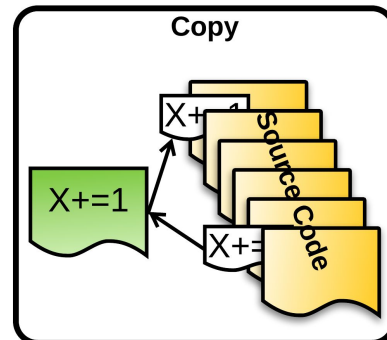
- Replace**
 - Copy + Delete
- Swap**
 - 2x Copy + 2x Delete



46

Copy

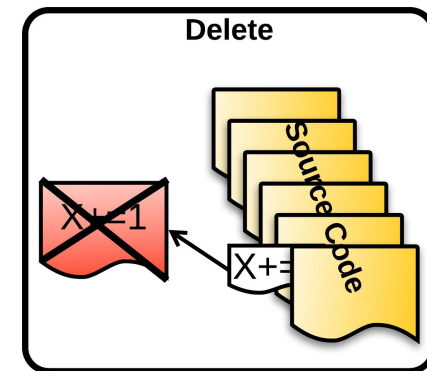
- CTRL+C => CTRL+V
- Applied to whole lines
- Some restrictions on what lines can be copied
 - Identified with regular expressions



47

Delete

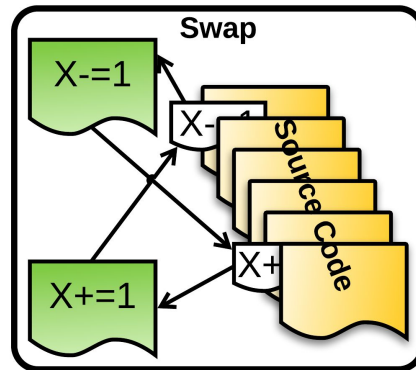
- Adds "#" to beginning of line
 - "Comment"
- Applied to whole lines
- Some restrictions on what lines can be commented out
 - Identified with regular expressions
- Can be reversed for previously deleted lines
 - "Uncomment"



48

Swap

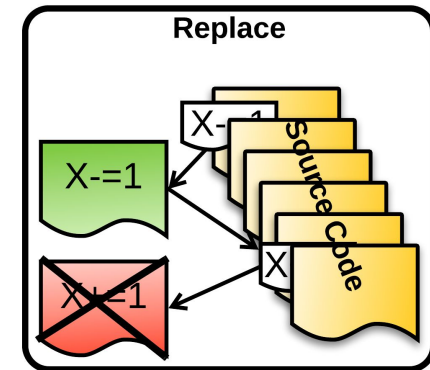
- Copies both lines above each other
- Then deletes the originals
- Applied to whole lines
 - Like for like



49

Replace

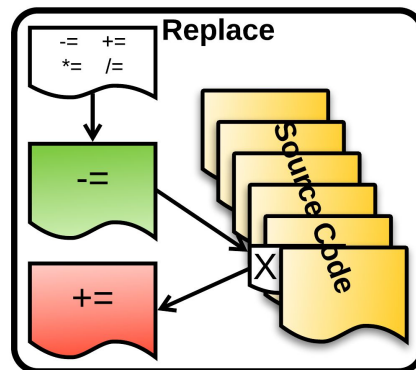
- Copies one line above another
- Then deletes that line



50

Replace -- extra

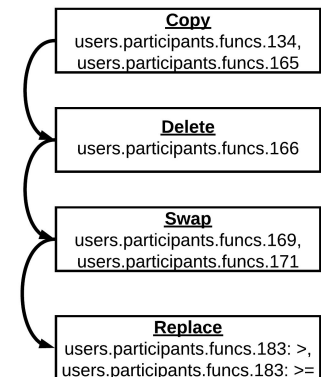
- Deep parameter tuning
- Operator specific replacement
 - and numbers too
- From a list of equivalent operators.



51

A list of edits makes a suggestion

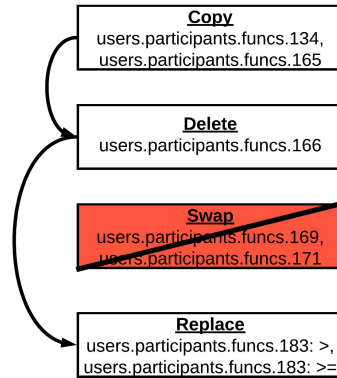
- Reads like a recipe
 - Step-by-step
- Automatically reduced
 - Delta debugging
- Scrutinised by the developer
 - Might change the recipe



52

A list of edits makes a suggestion

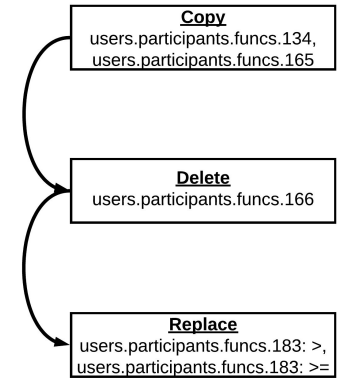
- Reads like a recipe
 - Step-by-step
- Automatically reduced
 - Delta debugging
- Scrutinised by the developer
 - Might change the recipe



53

A list of edits makes a suggestion

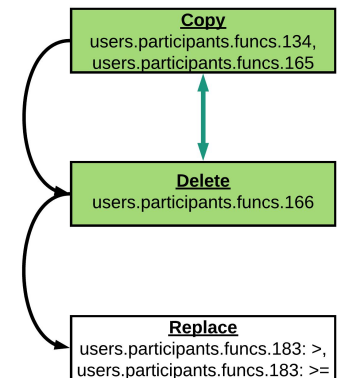
- Reads like a recipe
 - Step-by-step
- Automatically reduced
 - Delta debugging
- Scrutinised by the developer
 - Might change the recipe



54

A list of edits makes a suggestion

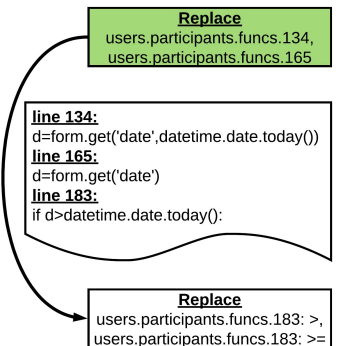
- Reads like a recipe
 - Step-by-step
- Automatically reduced
 - Delta debugging
- Scrutinised by the developer
 - Might change the recipe



55

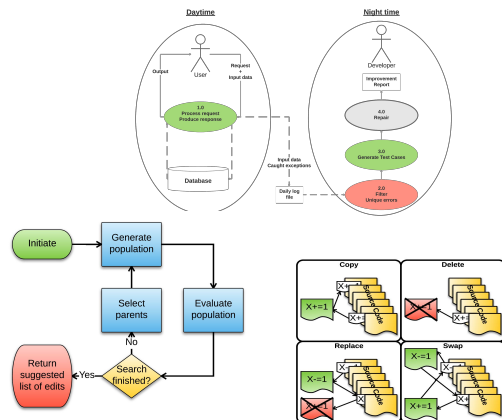
A list of edits makes a suggestion

- Reads like a recipe
 - Step-by-step
- Automatically reduced
 - Delta debugging
- Scrutinised by the developer
 - Might change the recipe



56

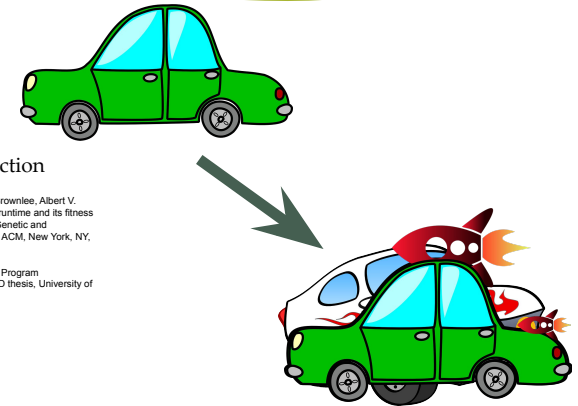
- Real-world example
- Catches inputs that produce crashes
- Line(-ish) based GI
 - 4 types of edits
- Overnight repair
- Developers are the gatekeepers



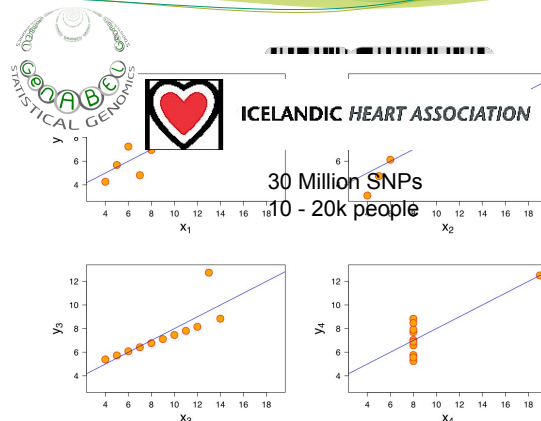
Faster

Saemundur O. Haraldsson, John R. Woodward, Alexander E. I. Brownlee, Albert V. Smith, and Vilmurður Gudnason. 2017. Genetic improvement of runtime and its fitness landscape in a bioinformatics application. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '17)*. ACM, New York, NY, USA, 1521-1528. DOI: <https://doi.org/10.1145/3067695.3082526>

S.O. Haraldsson. 2017. 'Genetic Improvement of Software: From Program Landscapes to the Automatic Improvement of a Live System', PhD thesis, University of Stirling, Stirling. <http://hdl.handle.net/1893/26007>

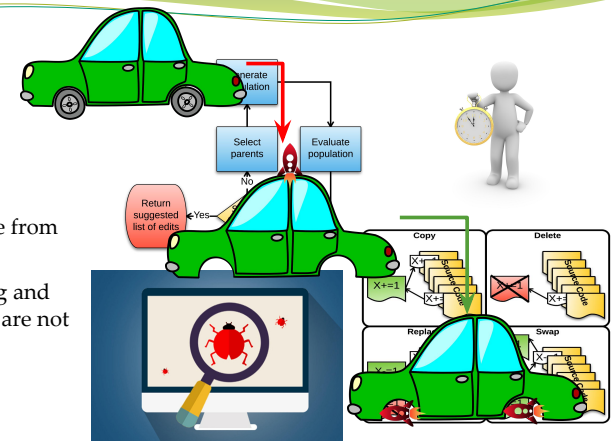


- A tool for Genome Wide Association studies.
- Collection of functions for regression models
- Written in C and C++
 - 8k LOC
 - 31 files
- Typical execution time around 8-12 hours



54

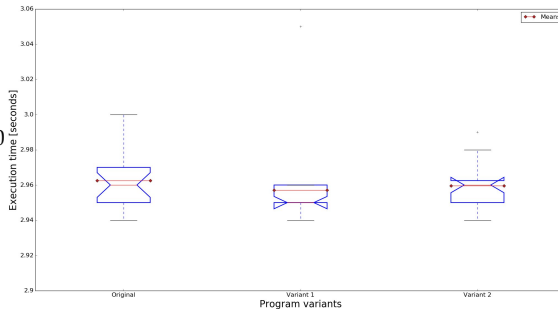
- Same as before
- Except for the evaluation
- Mean CPU time from 20 executions
- None compiling and failing variants are not discarded



60

Results

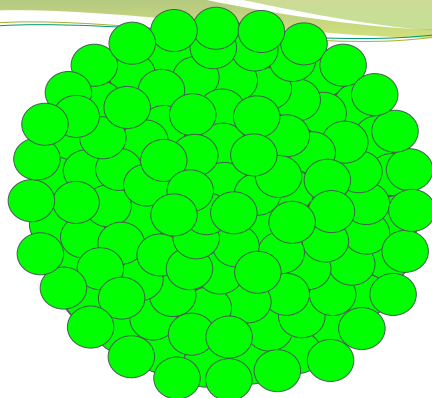
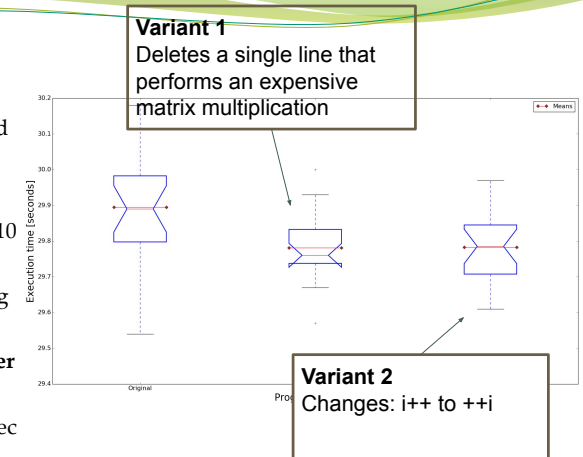
- 2 good variants found early on
 - < a second faster
 - Generations 5 and 10
- **Not** statistically significant on training dataset



61

Results

- 2 good variants found early on
 - < a second faster
 - Generations 5 and 10
- Not statistically significant on training dataset
- **Significant on a larger dataset**
 - Still, only about 1 sec faster



● Gained improvement per execution

63

Better predictions

And even more examples of GI in action

S. O. Haraldsson, R. D. Brynjólfsson, J. R. Woodward, K. Siggeirsdóttir and V. Gudnason, "The use of predictive models in dynamic treatment planning," 2017 IEEE Symposium on Computers and Communications (ISCC), Heraklion, 2017, pp. 242-247. DOI: <https://doi.org/10.1109/ISCC.2017.8024536>

S. O. Haraldsson, R. D. Brynjólfsson, V. Gudnason, K. Tomasson and K. Siggeirsdóttir, "Predicting changes in quality of life for patients in vocational rehabilitation," 2018 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS), Rhodes, 2018, pp. 1-8. DOI: <https://doi.org/10.1109/EAIS.2018.8397182>

Siggeirsdóttir, K., Brynjólfsson, R.D., Haraldsson, S.O., Vidar, S., Gudmundsson, E.G., Brynjólfsson, J.H., Jonsson, H., Hjaltason, O. and Gudnason, V., 2016. Determinants of outcome of vocational rehabilitation. Work, 55(3), pp.577-583. DOI: <https://doi.org/10.3233/WOR-162436>

S.O. Haraldsson, 2017. 'Genetic Improvement of Software: From Program Landscapes to the Automatic Improvement of a Live System', PhD thesis, University of Stirling, Stirling. <http://hdl.handle.net/1893/26097>



64

Dynamic updates to a prediction tool

- Used by Janus Rehabilitation
 - Since June 2016
 - Consulted in all team meetings
- Updated whenever there are new information
 - No developer as gatekeeper
- Target software is the updating script
 - Small python file

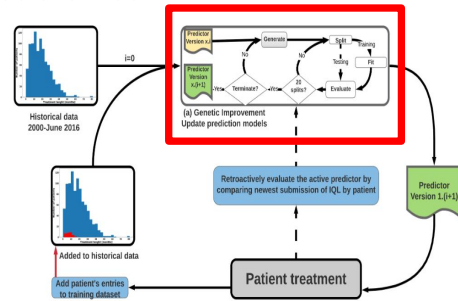


Fig. 1. The cycle of providing treatment, collecting data and updating the predictive models. This cycle is simulated from June 2016 until December 2017 of the simulation

65

The predictions

- Vocational rehabilitation outcome
 - Updated on every patient's discharge
 - Successful / Unsuccessful
 - Dropout
 - Length
- Next measurement of Icelandic Health-related Quality of Life (IQL)
 - Updated on every submission of questionnaire
 - 12 categories
 - Measured every 3-6 months

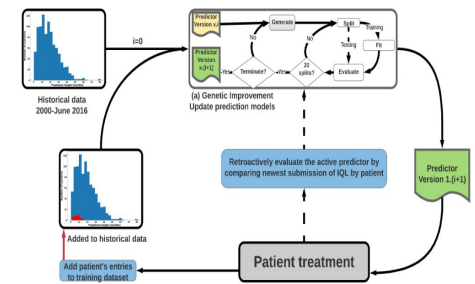
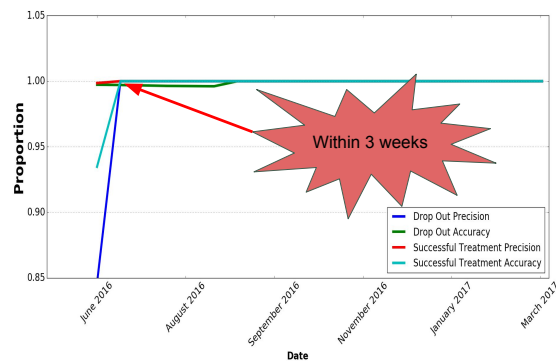


Fig. 1. The cycle of providing treatment, collecting data and updating the predictive models. This cycle is simulated from June 2016 until December 2017 of the simulation

66

Predicting the outcome

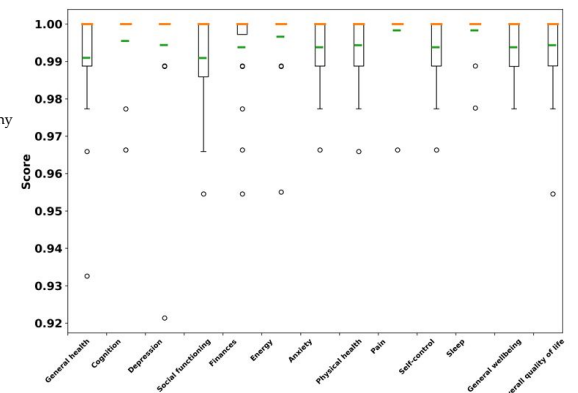
- Implemented in June 2016
 - Forgotten about for 10 months
- 72 updates over the 10 month period
 - Reached maximum accuracy early
- All predictions are for events that had not occurred.
 - Real people
 - Real events



67

Predicting the IQL

- Simulation
 - Bootstrapped accuracy distribution
- Never under 92% accuracy in any IQL subcategory
- Mean accuracy over 99%



Overview

- Introduction
- Fixing Bugs and other examples
- **Noteworthy papers and issues**
- Getting involved
- Summary and Q&A

69

Improving CUDA DNA Analysis Software with Genetic Programming (2015)

W.B. Langdon , B.Y.H. Lam , J. Petke & M. Harman



A 50,000 line system

1. DNA sequencing
2. consisting of 8,000+ lines of code.
3. improved version is up to 3x faster
4. downloaded 1,000 times.
5. **Ported by IBM** to one of their super computers

Optimising Existing Software with Genetic Programming

William B. Langdon and Mark Harman

- Bowtie2, a DNA sequence alignment/sequence analysis tool
- Using Genetic Improvement, Harman and Langdon were capable of increasing performance 70x.

A Systematic Study of Automated Program Repair: Fixing 55 out of 105 Bugs for \$8 Each

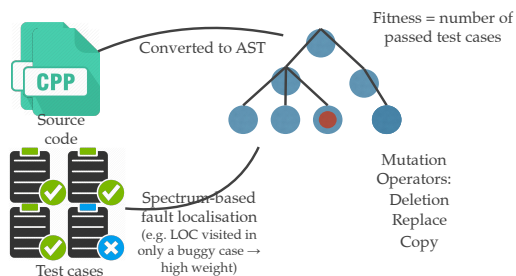
(2012)
Cited ~400 times

Claire Le Goues
Computer Science Department
University of Virginia
Charlottesville, VA
{legoues,mkd5m}@cs.virginia.edu

Michael Dewey-Vogt
Computer Science Department
University of New Mexico
Albuquerque, NM
forrest@cs.unm.edu

Stephanie Forrest
Computer Science Department
University of Virginia
Charlottesville, VA
forrest@cs.virginia.edu

Westley Weimer
Computer Science Department
University of Virginia
Charlottesville, VA
weimer@cs.virginia.edu



- Where an adequate test suite is provided, GenProg has been shown to fix real-world bugs
- It has inspired a variety of alternative frameworks, most of which claim to outperform GenProg

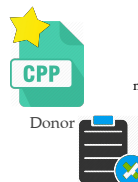
Automated Software Transplantation

(2015)

Earl T. Barr Mark Harman Yue Jia Alexandru Marginean Justyna Petke
CREST, University College London, Malet Place, London, WC1E 6BT, UK
{e.barr,m.harman,yue.jia,alexandru.marginean,j.petke}@ucl.ac.uk



Featured in:



Babel Pidgin: SBSE Can Grow and Graft Entirely New Functionality into a Real World System

(2014)

Mark Harman, Yue Jia, and William B. Langdon
University College London, CREST centre, UK

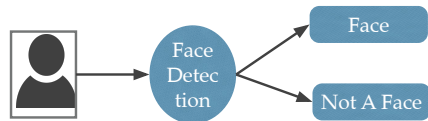


English to Korean;
English to Portuguese



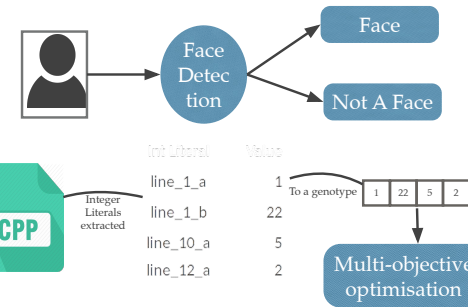
Deep Parameter Optimisation for Face Detection Using the Viola-Jones Algorithm in OpenCV

Bobby R. Bruce^{1(ES)}, Jonathan M. Aitken^{2(ES)}, and Justyna Petke^{1(ES)}



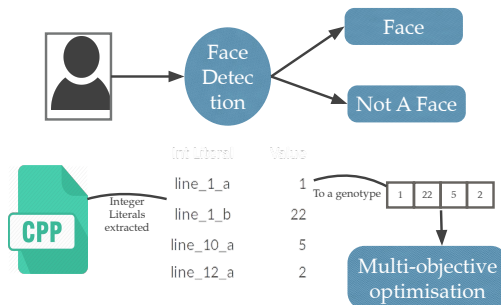
Deep Parameter Optimisation for Face Detection Using the Viola-Jones Algorithm in OpenCV

Bobby R. Bruce^{1(ES)}, Jonathan M. Aitken^{2(ES)}, and Justyna Petke^{1(ES)}

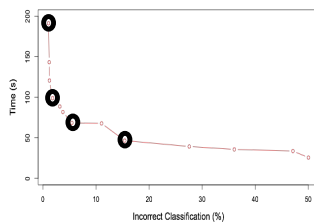


Deep Parameter Optimisation for Face Detection Using the Viola-Jones Algorithm in OpenCV

Bobby R. Bruce^{1(ES)}, Jonathan M. Aitken^{2(ES)}, and Justyna Petke^{1(ES)}



Original: 191s, 1.04% inaccuracy
 99s (48% decrease), 1.8% inaccuracy
 68s (64% decrease), 5.4% inaccuracy
 46s (76% decrease), 15.4% inaccuracy

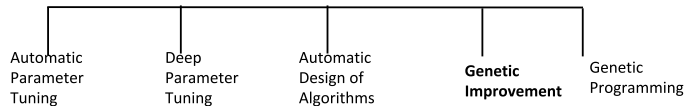


Phd Theses

- David R. White
- Andrea Accuri
- Bobby R. Brown
- Sæmundur Ó. Haraldsson
- Mahmoud R. Bokhari
- And many more to come...

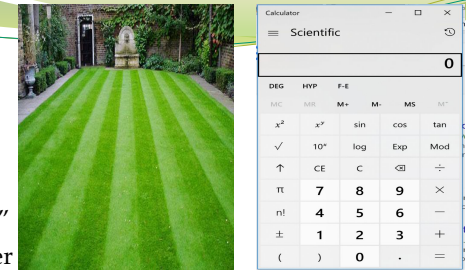
Relationship to other fields

- Optimization/machine learning - OVERFITTING (or: specialisation?)
("Is the cure worse than the disease?" Smith et al. FSE 2015)
- Genetic Programming and Metaheuristics
- the automatic design of algorithms
- Automatic parameter tuning/deep parameter tuning/GI



GI & Benchmarking

1. GP suffered a "midlife crisis"
2. Toy problem e.g. lawnmower
3. Genetic Programming Needs Better Benchmarks [White et al.]
4. Machine Learning that Matter [Wagstaff 2012] what is 1% meaning
5. *Is Software Engineering the best benchmark for GP?*
6. Do we have a stable set of benchmarks for GI?
(for program repair: <http://program-repair.org/benchmarks.html>)
7. Benchmarking is more complex (noise, hardware, prog lang, ...)



Measuring Energy

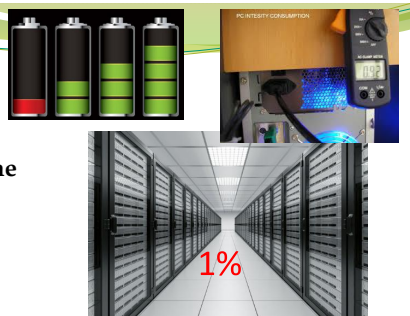
- computational energy consumption growing importance, particularly at the extremes (i.e., mobile devices and datacentres).

one line = one unit

simulate (runtime/system calls/) Tools
Opacitor, PowerGauge

read battery indicator

physically measure (e.g. see Bokhari et al.)



GI@GECCO'17 Deep Parameter Optimisation on Android Smartphones for Energy Minimisation - A Tale of Woe and a Proof-of-Concept

CEC 2019 Mind the gap - a distributed framework for enabling energy optimisation on modern smart-phones in the presence of noise, drift, and statistical insignificance [#19776]

Energy Tradeoff

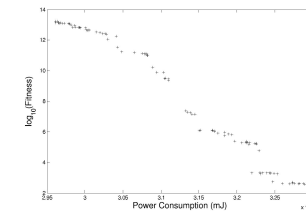


Figure 3: Archive at Generation 249, Experiment 1. The graph shows the trade-offs made by programs within the archive, between total power consumption and error. For both objectives, lower values are better.

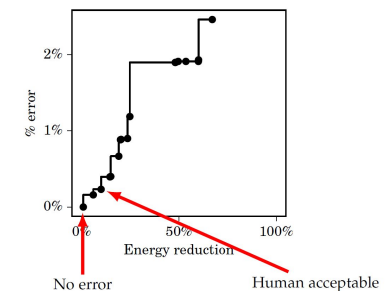


Fig. 1: Pareto frontier for the blender (car) benchmark. The X axis indicates percentage energy reduction and the Y axis indicates percentage error. The point in the lower-left has no error and corresponds to a 1% energy savings. The point in the lower-right was judged to be human acceptable and corresponds to a 10% energy savings.

GENETIC PROGRAMMING AND EVOLVABLE MACHINES

Editor-in-Chief: Lee Spector
Founding Editor: Wolfgang Banzhaf

The 62nd CREST Open Workshop - Automated Program Repair and Genetic Improvement
Date: 20th and 21st January 2020
Venue: George Fox room Friends House, 173-177 Euston Road, London, NW1 2BJ

video

activities

GI @ GECCO 2020
International Workshop on Automatic Software Optimisation
Ninth GI Workshop, 2020 (now running online!)

GI @ ICSE 2020
An International Workshop on the Repair and Optimisation of Software using Computational Search

International Workshop on Automatic Software Optimisation
Seventh edition of GI @ GECCO 2019 in Prague, Czech Republic

The 6th International Workshop on Genetic Improvement @ ICSE 2019

2nd International Summer School
We are proud to announce the Second International Summer School on Search-Based Software Engineering (SS-BSSE 2019).

January 2018
SCHLOSS DAGSTUHL
Leibniz-Zentrum für Informatik



Source of Genetic Material

1. the program being improved,
2. a different program written in the same language (Petke: MiniSAT competition),
3. a piece of code generated from scratch (GP),
4. different programming language other than the software to be improved.

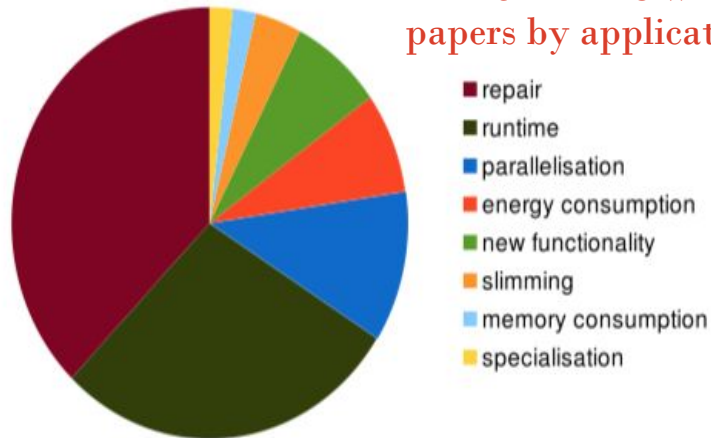
J. Petke, M. Harman, W. B. Langdon, and W. Weimer, "Using genetic improvement and code transplants to specialise a C++ program to a problem class," in *European Conf. on Genetic Programming EuroGP*, ser. LNCS, vol. 8599. Springer, 2014, pp. 137-149.

Theory

- Hard!
- NFL not really valid for GP, and therefore GI.
 - Why – because many programs share same functionality.

=> GI will remain empirical for years to come

BREAKDOWN papers by application



Grant Writing

- A grant about GP (0%)

VS

- A grant about GI. (100%)



Websites Genetic Improvement Workshop

An International Workshop on the Repair and Optimisation of Software using Computational Search

- <http://geneticimprovementofsoftware.com/>
- [https://en.wikipedia.org/wiki/Genetic_improvement_\(computer_science\)](https://en.wikipedia.org/wiki/Genetic_improvement_(computer_science))
- <http://www.davidrwhite.co.uk/>
- <http://daase.cs.ucl.ac.uk/>
- <http://crest.cs.ucl.ac.uk/publications/>
- <https://clairelegoues.com/blog/>
- <https://cs.adelaide.edu.au/~optlog/research/software.php>

Google Scholar

label:genetic_improvement



Starting point – POP science, GIN, Survey

IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION

Genetic Improvement of Software: a Comprehensive Survey (2017)

Justyna Petke, Saemundur O. Haraldsson, Mark Harman, William B. Langdon, David R. White, and John R. Woodward

A Survey of Genetic Improvement Search Spaces

GI@GECCO'19

Justyna Petke
Department of Computer Science
University College London
London, UK
j.petke@ucl.ac.uk

Alexander E.I. Brownlee
Computing Science and Mathematics
University of Stirling
Stirling, UK
shr@cs.stir.ac.uk

Brad Alexander
School of Computer Science
University of Adelaide
Adelaide, Australia
brad@cs.adelaide.edu.au

Markus Wagner
School of Computer Science
University of Adelaide
Adelaide, Australia
markus.wagner@adelaide.edu.au

Earl T. Barr
Department of Computer Science
University College London
London, UK
e.barr@ucl.ac.uk

David R. White
Department of Computer Science
The University of Sheffield
Sheffield, UK
d.r.white@sheffield.ac.uk

Overview

- Introduction
- Fixing Bugs and other examples
- Noteworthy papers and issues
- **Getting involved**
- Summary and Q&A

89

GI in No Time

David R. White
UCL, London, UK
david.r.white@ucl.ac.uk

(Inaugural paper at
GI@GECCO 2017)

Get involved with GI in No time - or GIN

Available at
<https://github.com/gintool/gin>



<http://www.davidrwhite.co.uk/>



v2.0 published in June 2020
"Gin: Genetic Improvement Research
Made Easy" (GECCO 2020)

90

"Stupidly simple"

GIN



ECJ



<https://cs.gmu.edu/~eclab/projects/ecj/>

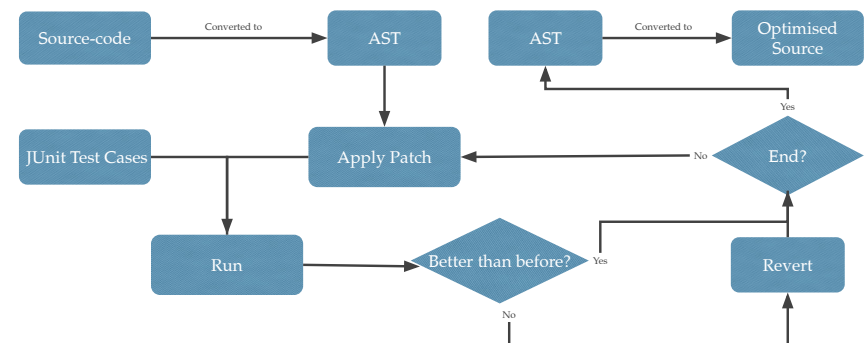
Genetic Improvement

- Many success stories
- ...however, these typically need a GI expert in the loop
- What's needed is a more systematic approach
- A toolkit to enable experimentation

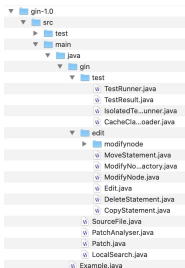
Gin's Goals

- Remove *incidental* difficulties of GI for research and teaching
- Enable focus on general questions
- Provide a central tool for the community
- Support more than bug-fixing: non-functional properties
- Work on open-source software projects out-of-the-box

Vanilla GIN: Neighbourhood search

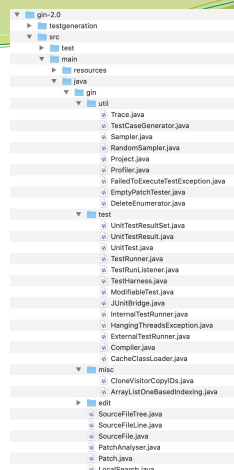


Vanilla GIN Version 1.0



Vanilla GIN Version 2.0:

gradle/maven support,
various types of edits,
profiler to find "hot" methods,
various samplers, ...



The inaugural paper
official V2.0 released on 12 June 2019:
<https://github.com/gintool/gin/releases>

Gin: Genetic Improvement Research Made Easy

Alexander E.I. Brownlee
Computing Science and Mathematics
University of Stirling
Stirling, UK
s.barr@stir.ac.uk

Earl T. Barr
Department of Computer Science
University of Stirling
Stirling, UK
e.barr@stir.ac.uk

Justyna Petke
Department of Computer Science
University of Stirling
Stirling, UK
j.petke@stir.ac.uk

Markus Wagner
School of Computer Science
University of Stirling
Stirling, UK
markus.wagner@stir.ac.uk

Brad Alexander
School of Computer Science
University of Stirling
Stirling, UK
brad@cs.stir.ac.uk

David R. White
Department of Computer Science
The University of Sheffield
Sheffield, UK
d.r.white@sheffield.ac.uk

ABSTRACT

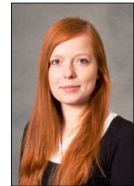
Genetic improvement (GI) is a young field of research on the cusp of transforming software development. GI uses search to improve existing software. Researchers have already shown that GI can improve human-written code, ranging from program repair to optimising run-time, from reducing energy-consumption to the transmutation of new functionality. Much remains to be done. The cost of re-implementing GI to investigate new approaches is hindering progress. Therefore, we present Gin, an extensible and modifiable

1 INTRODUCTION

Genetic improvement (GI) is a young field of software engineering research that uses search to improve existing software. GI aims to improve both functional, notably bug fixing, and non-functional properties of software, such as runtime or energy consumption. The intersection of automated program repair (APR) and GI has had the greatest impact to date, from the release of the GI-based tool GenProg [27] to successful integration of APR into commercial development processes [19, 20]. Non-functional improvement (NFI) is



Bradley Alexander



Justyna Petke



Earl T.
Barr



Markus Wagner

Also uses GIN in teaching since 2017
<https://tinyurl.com/giassignment>



Sandy Brownlee

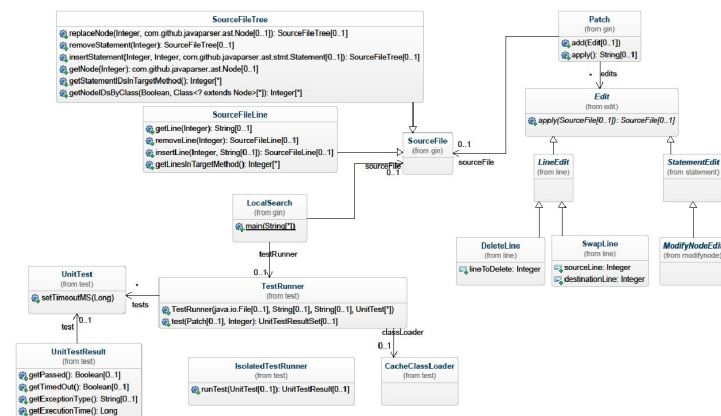


David R. White

Gin Design

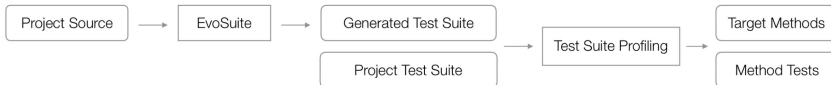


Gin Core Classes



Gin Pipelines

Preprocessing



Search Space Analysis



Edits

- Edits are single changes to source code
 - Building blocks of a repair
 - Combined into Patches
 - Question: actually, what scale might an *edit* be?
- Gin supports edits at:
 - line level (Langdon) - delete/replace/copy/swap/move
 - statement level (GenProg) - delete/replace/copy/swap/move
 - constrained (matched) statement - replace/swap
 - micro edits
 - binary & unary operator replacement (OR \leftrightarrow AND) ($++ \leftrightarrow --$)
 - reorder Boolean expressions ($X \&\& Y \leftrightarrow Y \&\& X$)

Edits

- We provide many wrappers to make your life easier, so that you can focus on higher-level tasks:
 - “Tell me which lines are eligible for deletion in this method”
 - “Delete this line”
 - “Give me all the for loop conditions in this method”
 - And many more...

Example edits

```
1 public class ReplaceStatement extends StatementEdit {
2
3     public int sourceID;
4     public int destinationID;
5
6     public ReplaceStatement(SourceFileTree sf, Random r) {
7         sourceID = sf.getRandomStatementID(false, r);
8         destinationID = sf.getRandomStatementID(true, r);
9     }
10
11     public SourceFile apply(SourceFileTree sf) {
12         Statement source = sf.getStatement(sourceID);
13         Statement dest = sf.getStatement(destinationID);
14         return sf.replaceNode(dest, source.clone());
15     }
16
17 }
```

Example edits

```
1 public class MatchedReplaceStatement extends
2     ReplaceStatement {
3     public MatchedReplaceStatement(SourceFileTree sf,
4         Random r) {
5         super(0, 0);
6         destinationID = sf.getRandomStatementID(true, r);
7         sourceID = sf.getRandomNodeID(false,
8             sf.getStatement(destinationID).getClass(), r);
9     }
10 }
```

Patch Evaluation

Gin invokes test cases via JUnit...

Tracks:

- compile success;
- run-time errors, exception types
- actual & expected outcomes
- timing: wall-clock and CPU time

```
UnitTests() ut = {
    new UnitTest("TriangleTest", "testInvalidTriangles"),
    new UnitTest("TriangleTest", "testEqualateralTriangles"),
    new UnitTest("TriangleTest", "testIsoscelesTriangles"),
    new UnitTest("TriangleTest", "testScaleneTriangles")
};

UnitTests.defaultTimeoutMS = 10000;
int reps = 1;

SourceFileTree sf = new SourceFileTree("examples/triangle/Triangle.java",
    Collections.singletonList("classifyTriangle(int,int,int)"));

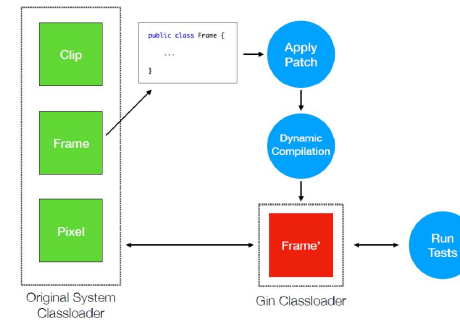
InternalTestRunner tr = new InternalTestRunner("TriangleTest",
    "examples/triangle", Arrays.asList(ut));

// Start with the empty patch
Patch patch = new Patch(sf);

// Run empty patch and log
UnitTestResultSet rs = tr.runTests(patch, reps);

boolean compiled = rs.getCleanCompile();
boolean testTimedOut = rs.getResults().get(0).getTimedOut();
long test0ExecutionTime = rs.getResults().get(0).getExecutionTime();
String test0ExceptionMessage = rs.getResults().get(0).getExceptionMessage();
```

An analogy: video editing.
Here: Gin Compiles and Reloads
on-the-fly



Note: If you prefer to use the more "traditional" way of writing the file to disk first - e.g., due to integration of Gin into other pipelines - then you can use a command-line flag to do so.

Sampling

- Included samplers:
 - EmptyPatchTester
 - RandomSampler
 - DeleteEnumerator
 - LocalSearch

- Possible Questions:

- What is the effectiveness of a given edit type for fixing a category of bug?
- How robust is the space of single-line edits, modulo the given test suite?
- ...

```
1 public static void main(String[] args) {
2
3     UnitTests() ut = {
4         new UnitTest("TriangleTest", "testInvalidTriangles"),
5         ...
6     };
7
8     int reps = 1;
9
10    SourceFileTree sf = new SourceFileTree(
11        "examples/simple/Triangle.java",
12        Collections.singletonList(
13            "classifyTriangle(int,int,int)"));
14
15    TestRunner tr = new TestRunner(
16        new File("examples/simple"), "Triangle",
17        "examples/simple", Arrays.asList(ut));
18
19    // Start with the empty patch
20    Patch patch = new Patch(sf);
21
22    // Run empty patch and log
23    UnitTestResultSet rs = tr.test(patch, reps);
24    writeResults(rs, 0);
25
26    int patchCount = 0;
27    for (int id : sf.getStatementIdsInTargetMethod()) {
28        patchCount++;
29        patch = new Patch(sf);
30        patch.add(new DeleteStatement(sf.getFilename(), id));
31
32        rs = tr.test(patch, reps);
33        writeResults(rs, patchCount);
34    }
35 }
```

Sampling

The following is one really wide output file:

IndexPath PatchSize Patch

```
1 1 | gin.edit.statement.SwapStatement ./src/main/java/org/jcodec/codecs/vpx/VPXBitstream.java:752 <-> ./src/main/java/org/jcodec/codecs/vpx/VPXBitstream.java:884 |
2 1 | gin.edit.statement.ReplaceStatement ./src/main/java/org/jcodec/codecs/prores/ProresEncoder.java:2310 -> ./src/main/java/org/jcodec/codecs/prores/ProresEncoder.java:1185 |
3 1 | gin.edit.statement.CopyStatement ./src/main/java/org/jcodec/codecs/containers/mp4/boxes/Box.java:514 -> ./src/main/java/org/jcodec/codecs/containers/mp4/boxes/Box.java:110 |
```

TestTimedOut	TestExceptionType	TestExceptionMessage	AssertionExpectedValue	AssertionActualValue
FALSE	java.lang.AssertionError	expected:<255> but was:<207>	255	207
FALSE	N/A	N/A	N/A	N/A
FALSE	N/A	N/A	N/A	N/A

MethodIndex	TestIndex	UnitTest	RepNumber	PatchValid	PatchCompiled	TestPassed	TestExecutionTime(ns)	TestCPUTime(ns)
152	1	org.jcodec.codecs.vpx.TestCoeffEncoder.testCoeffDCTU []	0	TRUE	TRUE	FALSE	2853708	1535633
189	1	org.jcodec.codecs.prores.ProresEncoderTest.testWholeThing []	0	TRUE	FALSE	FALSE	0	0
184	1	org.jcodec.codecs.containers.mp4.boxes.BoxTest.testReadWriteCreate []	0	TRUE	FALSE	FALSE	0	0

Local search

```
1 private Patch search() {
2     // start with the empty patch
3     Patch bestPatch = new Patch(sourceFile);
4     long bestTime = testRunner.test(bestPatch, 10).
5         totalExecutionTime();
6
7     for (int step = 1; step <= NUM_STEPS; step++) {
8         Patch neighbour = neighbour(bestPatch, rng);
9         UnitTestResultSet rs = testRunner.test(neighbour,
10             10);
11         if (rs.isValidPatch() && rs.getCleanCompile() &&
12             rs.allTestsSuccessful() &&
13             rs.totalExecutionTime() < bestTime) {
14             bestPatch = neighbour;
15             bestTime = rs.totalExecutionTime();
16         }
17     }
18     return bestPatch;
19 }
20
21 public Patch neighbour(Patch patch, Random rng) {
22     Patch neighbour = patch.clone();
23
24     if (neighbour.size() > 0 && rng.nextFloat() > 0.5) {
25         neighbour.remove(rng.nextInt(neighbour.size()));
26     } else {
27         neighbour.addRandomEdit(rng, allowableEditTypes);
28     }
29     return neighbour;
30 }
```

Local search, output

```
-bash-4.1$ java -jar build/ging.jar gin.LocalSearch -filename examples/triangle/Triangle.java -m "classifyTriangle(int, int, int)"
2020-04-10 04:36:41 gin.LocalSearch.search() INFO: LocalSearch on file: examples/triangle/Triangle.java method: classifyTriangle(int, int, int)
2020-04-10 04:36:44 gin.test.InternalTestRunner.runSingleTest() WARNING: Possible hanging threads remain after test
2020-04-10 04:36:59 gin.LocalSearch.search() INFO: Original execution time: 166691219ms
2020-04-10 04:36:59 gin.LocalSearch.search() INFO: Step: 1, Patch: | gin.edit.line.ReplaceLine examples/triangle/Triangle.java:5 -> examples/triangle/Triangle.java:23
|, Failed to compile
2020-04-10 04:36:59 gin.LocalSearch.search() INFO: Step: 2, Patch: | gin.edit.line.DeleteLine examples/triangle/Triangle.java:36 |, Failed to compile
2020-04-10 04:36:59 gin.LocalSearch.search() INFO: Step: 3, Patch: | gin.edit.line.DeleteLine examples/triangle/Triangle.java:19 |, Failed to compile
2020-04-10 04:36:59 gin.LocalSearch.search() INFO: Step: 4, Patch: | gin.edit.line.DeleteLine examples/triangle/Triangle.java:2 |Failed to pass all tests
2020-04-10 04:36:59 gin.LocalSearch.search() INFO: Step: 5, Patch: | gin.edit.line.ReplaceLine examples/triangle/Triangle.java:38 -> examples/triangle/Triangle.java:31
|, Failed to compile
2020-04-10 04:36:59 gin.LocalSearch.search() INFO: Step: 6, Patch: | gin.edit.line.DeleteLine examples/triangle/Triangle.java:17 |, Failed to compile
2020-04-10 04:37:00 gin.LocalSearch.search() INFO: Step: 7, Patch: | gin.edit.line.CopyLine examples/triangle/Triangle.java:34 -> examples/triangle/Triangle.java:13 |,
Failed to compile
2020-04-10 04:37:00 gin.test.InternalTestRunner.runSingleTest() WARNING: Possible hanging threads remain after test
2020-04-10 04:37:00 gin.test.InternalTestRunner.runSingleTest() WARNING: Possible hanging threads remain after test
2020-04-10 04:37:00 gin.LocalSearch.search() INFO: Step: 8, Patch: | gin.edit.line.SwapLine examples/triangle/Triangle.java:27 <-> examples/triangle/Triangle.java:10 |
Failed to pass all tests
...
2020-04-10 04:36:26 gin.LocalSearch.search() INFO: Step: 96, Patch: | gin.edit.line.DeleteLine examples/triangle/Triangle.java:10 | gin.edit.line.SwapLine
examples/triangle/Triangle.java:8 <-> examples/triangle/Triangle.java:14 |, Failed to compile
2020-04-10 04:36:28 gin.LocalSearch.search() INFO: Step: 97, Patch: |, Time: 1667322167ns
2020-04-10 04:36:28 gin.LocalSearch.search() INFO: Step: 98, Patch: | gin.edit.line.DeleteLine examples/triangle/Triangle.java:10 | gin.edit.line.CopyLine
examples/triangle/Triangle.java:51 -> examples/triangle/Triangle.java:26 |, Failed to compile
2020-04-10 04:36:29 gin.LocalSearch.search() INFO: Step: 99, Patch: |, Time: 166831018ns
2020-04-10 04:36:29 gin.LocalSearch.search() INFO: Step: 100, Patch: | gin.edit.line.DeleteLine examples/triangle/Triangle.java:10 | gin.edit.line.SwapLine
examples/triangle/Triangle.java:19 <-> examples/triangle/Triangle.java:27 (ns), New best time: 38744892 (ns), Speedup (x): 97.64, Patch: | gin.edit.line.DeleteLine
examples/triangle/Triangle.java:10 |
```

Local search:
What did we actually optimise here?

```
-bash-4.1$ cat examples/triangle/Triangle.java
public class Triangle {
    static final int INVALID = 0;
    static final int SCALE = 1;
    static final int EQUALATERAL = 2;
    static final int ISOSCELES = 3;

    public static int classifyTriangle(int a, int b, int c) {
        delay();

        // Sort the sides so that a <= b <= c
        if (a > b) {
            int tmp = a;
            a = b;
            b = tmp;
        }
        if (a > c) {
            int tmp = a;
            a = c;
            c = tmp;
        }
        if (b > c) {
            int tmp = b;
            b = c;
            c = tmp;
        }

        if (a + b <= c) {
            return INVALID;
        } else if (a == b && b == c) {
            return EQUALATERAL;
        } else if (a == b || b == c) {
            return ISOSCELES;
        } else {
            return SCALE;
        }

        private static void delay() {
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {}
        }
    }
}
```

The
problematic
line was
deleted.

```
-bash-4.1$ cat examples/triangle/Triangle.java.optimised
public class Triangle {
    static final int INVALID = 0;
    static final int SCALE = 1;
    static final int EQUALATERAL = 2;
    static final int ISOSCELES = 3;

    public static int classifyTriangle(int a, int b, int c) {

        // Sort the sides so that a <= b <= c
        if (a > b) {
            int tmp = a;
            a = b;
            b = tmp;
        }
        if (a > c) {
            int tmp = a;
            a = c;
            c = tmp;
        }
        if (b > c) {
            int tmp = b;
            b = c;
            c = tmp;
        }

        if (a + b <= c) {
            return INVALID;
        } else if (a == b && b == c) {
            return EQUALATERAL;
        } else if (a == b || b == c) {
            return ISOSCELES;
        } else {
            return SCALE;
        }

        private static void delay() {
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {}
        }
    }
}
```

Generating tests and Profiling

Generate new test cases

```
java -cp build/ging.jar gin.util.TestCaseGenerator
-projectDir examples/maven-simple -projectName my-app
-classNames com.mycompany.app.App -generateTests
```

Profile a test suite

```
java -cp build/ging.jar gin.util.Profiler -p my-app
-d examples/maven-simple/
```

Results written to profiler_output.csv.

Build tool integration

- Maven and Gradle API documentation is sparse!
 - And many projects seem to break conventions about paths, resources etc.
- **Project** class wraps most of what we have learned
 - provide the classpath for a project
 - find a particular source file within a project's file hierarchy
 - provide a standard method signature for a given method
 - provide a list of project tests
 - run a unit test given its name
- Gin can infer the necessary classpath and dependencies for running unit tests from a Maven or Gradle project, or these can be specified manually
- Maven projects can be updated automatically with new unit tests from *EvoSuite*

Examples with jCodec (maven project)

```
• Profiler
projectnameforgin='jcodec';
java -Dtinylog.level=trace -cp ../../ginfork/build/gin.jar gin.util.Profiler
-h ~/.sdkman/candidates/maven/current/ -p $projectnameforgin -d .
-o $projectnameforgin.Profiler_output.csv -r 1
```

Examples with jCodec (maven project)

```
• Profiler
projectnameforgin='jcodec';
java -Dtinylog.level=trace -cp ../../ginfork/build/gin.jar gin.util.Profiler
-h ~/.sdkman/candidates/maven/current/ -p $projectnameforgin -d .
-o $projectnameforgin.Profiler_output.csv -r 1
```

Examples with jCodec (maven project)

```
• Profiler
projectnameforgin='jcodec';
java -Dtinylog.level=trace -cp ../../ginfork/build/gin.jar gin.util.Profiler
-h ~/.sdkman/candidates/maven/current/ -p $projectnameforgin -d .
-o $projectnameforgin.Profiler_output.csv -r 1

• EmptyPatchTester
projectnameforgin='jcodec';
java -Dtinylog.level=trace -cp ../../ginfork/build/gin.jar gin.util.EmptyPatchTester -h
~/.sdkman/candidates/maven/current/ -p $projectnameforgin -d .
-m $projectnameforgin.Profiler_output.csv
-o $projectnameforgin.EmptyPatchTester_output.csv
```

Examples with jCodec (maven project)

- Profiler

```
projectnameforgin='jcodec';
java -Dtinylog.level=trace -cp ../../ginfork/build/gin.jar gin.util.Profiler
-h ~/.sdkman/candidates/maven/current/ -p $projectnameforgin -d .
-o $projectnameforgin.Profiler_output.csv -r 1
```
- EmptyPatchTester

```
projectnameforgin='jcodec';
java -Dtinylog.level=trace -cp ../../ginfork/build/gin.jar gin.util.EmptyPatchTester -h
~/.sdkman/candidates/maven/current/ -p $projectnameforgin -d .
-m $projectnameforgin.Profiler_output.csv
-o $projectnameforgin.EmptyPatchTester_output.csv
```
- PatchSampler

```
projectnameforgin='jcodec';
java -Dtinylog.level=trace -cp ../../ginfork/build/gin.jar gin.util.PatchSampler
-h ~/.sdkman/candidates/maven/current/ -p $projectnameforgin -d .
-m $projectnameforgin.Profiler_output.csv
-o $projectnameforgin.PatchSampler_LINE_output.csv -editType LINE -patchNo 100
```
- Generate tests

Overview

- Introduction
- Fixing Bugs and other examples
- Noteworthy papers and issues
- Getting involved
- Summary and Q&A

Gin

- Available at <https://github.com/gintool/gin>

- The team actively uses Gin to push the GI boundaries, and quite a few papers are in the works.

- Open for contributions!
 - Particularly new edits and tools
 - <https://github.com/gintool/gin>
 - we'd like this to become the MiniSAT of GI

Watch 11 Star 21 Fork 7

Comments/questions: Sandy (Alexander E.I. Brownlee) sbr@cs.stir.ac.uk

Gin: Genetic Improvement Research Made Easy

Alexander E.I. Brownlee Computing Science and Mathematics University of Stirling Stirling, UK aeb@cs.stir.ac.uk	Justyna Petke Department of Computer Science University College London London, UK j.petke@ucl.ac.uk	Broad Alexander School of Computer Science University of Adelaide Adelaide, Australia broad@cs.adelaide.edu.au
Earl T. Barr Department of Computer Science University College London London, UK e.barr@ucl.ac.uk	Markus Wagner School of Computer Science University of Adelaide Adelaide, Australia markus.wagner@adelaide.edu.au	David R. White Department of Computer Science The University of Sheffield Sheffield, UK d.r.white@sheffield.ac.uk

Injecting Shortcuts for Faster Running Java Code

Alexander E.I. Brownlee Computing Science and Mathematics University of Stirling Stirling, UK aeb@cs.stir.ac.uk	Justyna Petke Department of Computer Science University College London London, UK j.petke@ucl.ac.uk	Anna F. Ruchon Computing Science and Mathematics University of Stirling Stirling, UK afr@cs.stir.ac.uk
--	--	---

Analysing Program Transformation Spaces for Genetic Improvement using Gin

Justyna Petke, Broad Alexander, Earl T. Barr, Alexander E.I. Brownlee, Markus Wagner, David R. White

Software Improvement with Gin: A Case Study

Justyna Petke¹ and Alexander E. I. Brownlee²

¹ University College London, London, UK

j.petke@ucl.ac.uk

² University of Stirling, Stirling, UK

aeb@cs.stir.ac.uk

Genetic Improvement vs Genetic Programming

1. Start from an existing program
2. BLOAT? – interpretation?
3. NO function / terminal set
4. Improvement of non-functional properties.
5. Easier to write grants
6. Different benchmarks.
7. Population of edits **NOT programs**.

PUTTING IT ALL TOGETHER

- Let's start with **existing programs**. Not like standard GP.
- Python vs C vs Java? Amenable to GI? Most popular
- Benchmarking ???
- Population of edits, not programs
- GP applied to real software
 - Large, loops, side-effect, modules,...
 - Non functional properties

Questions?

Saemundur (Saemi) Haraldsson <soh@cs.stir.ac.uk>
John Woodward <j.woodward@qmul.ac.uk>
Markus Wagner <markus.wagner@adelaide.edu.au>

122

Bibliography

- S.O. Haraldsson, John R. Woodward, Alexander E. I. Brownlee, and Kristin Siggeirsdottir. 2017. Fixing bugs in your sleep: how genetic improvement became an overnight success. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '17). ACM, New York, NY, USA, 1513-1520. DOI: <https://doi.org/10.1145/3067695.3082517>
- S. O. Haraldsson, J. R. Woodward and A. I. E. Brownlee, "The Use of Automatic Test Data Generation for Genetic Improvement in a Live System," 2017 IEEE/ACM 10th International Workshop on Search-Based Software Testing (SBST), Buenos Aires, 2017, pp. 28-31. DOI: <https://10.1109/SBST.2017.10>
- S.O. Haraldsson, 2017. 'Genetic Improvement of Software: From Program Landscapes to the Automatic Improvement of a Live System', PhD thesis, University of Stirling, Stirling. <http://hdl.handle.net/1893/26007>
- S.O. Haraldsson, John R. Woodward, Alexander E. I. Brownlee, Albert V. Smith, and Vilundur Gudnason. 2017. Genetic improvement of runtime and its fitness landscape in a bioinformatics application. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '17). ACM, New York, NY, USA, 1521-1528. DOI: <https://doi.org/10.1145/3067695.3082529>
- S.O. Haraldsson, 2017. 'Genetic Improvement of Software: From Program Landscapes to the Automatic Improvement of a Live System', PhD thesis, University of Stirling, Stirling. <http://hdl.handle.net/1893/26007>
- S. O. Haraldsson, R. D. Brynjólfsson, J. R. Woodward, K. Siggeirsdottir and V. Gudnason, "The use of predictive models in dynamic treatment planning," 2017 IEEE Symposium on Computers and Communications (ISCC), Heraklion, 2017, pp. 242-247. DOI: <https://10.1109/ISCC.2017.8024536>
- S. O. Haraldsson, R. D. Brynjólfsson, V. Gudnason, K. Tomasson and K. Siggeirsdottir, "Predicting changes in quality of life for patients in vocational rehabilitation," 2018 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS), Rhodes, 2018, pp. 1-8. DOI: <https://10.1109/EAIS.2018.8397182>
- Siggeirsdottir, K., Brynjólfsson, R.D., Haraldsson, S.O., Vidar, S., Gudmundsson, E.G., Brynjólfsson, J.H., Jonsson, H., Hjaltason, O. and Gudnason, V., 2016. Determinants of outcome of vocational rehabilitation. Work, 55(3), pp.577-583. DOI: <https://10.3233/WOR-162436>

123

- J. Pelke, B. Alexander, E.T. Barr, A.E.I. Brownlee, M. Wagner, and D.R. White, 2019. 'A survey of genetic improvement search spaces'. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '19). ACM, New York, NY, USA, 1715-1721. DOI: <https://doi.org/10.1145/3319619.3326870>
- A.E.I. Brownlee, J. Pelke, B. Alexander, E.T. Barr, M. Wagner, and D.R. White, 2019. 'Gin: genetic improvement research made easy'. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '19). ACM, New York, NY, USA, 985-993. DOI: <https://doi.org/10.1145/3321707.3321841>
- M.A. Bokhari, B. Alexander, and M. Wagner, 2019. 'In-vivo and offline optimisation of energy use in the presence of small energy signals: A case study on a popular Android library'. In Proceedings of the EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous '18). ACM, New York, NY, USA, 207-215. DOI: <https://doi.org/10.1145/3286978.3287014>
- M.A. Bokhari, B. Alexander, and M. Wagner, 2020. 'Towards Rigorous Validation of Energy Optimisation Experiments'. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '20). ACM, New York, NY, USA. URL: <https://arxiv.org/abs/2004.04500v1>
- M.A. Bokhari, B.R. Bruce, B. Alexander, and M. Wagner, 2017. 'Deep parameter optimisation on Android smartphones for energy minimisation: a tale of woe and a proof-of-concept'. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '17). ACM, New York, NY, USA, 1501-1508. URL: <https://doi.org/10.1145/3067695.3082519>
- M.A. Bokhari, L. Weng, M. Wagner, and B. Alexander, 2019. 'Mind the gap - a distributed framework for enabling energy optimisation on modern smart-phones in the presence of noise, drift, and statistical insignificance'. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC '19). IEEE, 1330-1337. DOI: <https://doi.org/10.1109/CEC.2019.8790246>
- A. Agrawal, T. Menzies, L. Minku, M. Wagner, and Z. Yu, 2020. 'Better software analytics via "DUO": Data mining algorithms using/used-by optimizers'. Empirical Software Engineering, Springer. Published 22 April 2020. DOI: <https://doi.org/10.1007/s10664-020-09808-9>
- V. Nair, A. Agrawal, J. Chen, W. Fu, G. Mathew, T. Menzies, L. Minku, M. Wagner, and Z. Yu, 2018. 'Data-driven search-based software engineering'. In Proceedings of the International Conference on Mining Software Repositories (MSR '18), ACM, New York, NY, USA, 341-352. DOI: <https://doi.org/10.1145/3196396.3196442>

124