Genetic Improvement: Taking real-world source code and improving it using computational search methods

Alexander Brownlee, Sæmundur Ó. Haraldsson, Markus Wagner, John R. Woodward

UNIVERSITY of STIRLING

MONASH University

Loughborough University

Latest version of slides at https://cs.stir.ac.uk/~sbr/files/GI_tutorial_GECCO_2024.pdf

GECCO '24 Companion, *July 14–18, 2024, Melbourne, VIC, Australia*

ACM ISBN 979-8-4007-0495-6/24/07.
https://doi.org/10.1145/3638530.3648418

Instructors

UNIVERSITY of STIRLING

- Saemundur O. Haraldsson is a Lecturer at the University of Stirling. He co-organised every version of this tutorial. He has multiple publications on Genetic Improvement, including two that have received best paper awards. Additionally, he co-authored the first comprehensive survey on GI 1 which was published in 2017. He has been invited to give talks on the subject in two Crest Open Workshops and for an industrial audience in Iceland. His PhD thesis (submitted in May 2017) details his work on the world's first live GI integration in an industrial application.
- Alexander (Sandy) Brownlee is a Senior Lecturer in the Division of Computing Science and Mathematics at the University of Stirling. His main topics of interest are in search-based optimisation methods and machine learning, with applications in civil engineering, transportation and SBSE. Within SBSE, he is interested in automated bug-fixing and improvement of non-functional properties such as run-time and energy consumption; how these different objectives interact with each other; and novel approaches to mutating code. He is also one of the developers of Gin, an open-source toolkit for experimentation with Genetic Improvement on real-world software projects.

Instructors

- Markus Wagner is an Associate Professor at the Department of Data Science and AI, Monash University, Australia. His research includes mathematical runtime analysis of heuristic optimization algorithms, theory-guided algorithm design, and applications of heuristic methods to software engineering and renewable energy production. He has led industry-funded projects by Google, Facebook, and other companies in defense and mining. He has authored about 200 articles and has attracted over AUD 10M in funding. His awards include one best poster, one best presentation, four best papers, one medal, and one Humies Gold Award.
- John R. Woodward is Head of Department at Loughborough University and previously led The Operational Research Group at Queen Mary University of London. He has also been a lecturer at the University of Stirling and the University of Nottingham. John holds a BSc in Theoretical Physics, an MSc in Cognitive Science, and a PhD in Computer Science from the University of Birmingham. His research interests include Automated Software Engineering, AI/Machine Learning, and Genetic Programming. He has over 50 publications and has given more than 50 talks at international conferences. John's experience spans industrial, military, educational, and academic settings, including employment with EDS, CERN, RAF, and three UK universities.

Overview

- **Introduction: why GI? John**
- **Basic principles: approaches, objectives John**
- **Challenges and open research questions Markus**
- **Case study: fixing bugs Saemi**
- **The human perspective Saemi**
- **Noteworthy papers, and connections to other topics Markus**
- **Demonstration: Gin Sandy**
- **Summary and Q&A John**

Overview

- **Introduction: why GI? And basic principles**
- **Challenges and open research questions**
- **Case study: fixing bugs**
- **The human perspective**
- **Noteworthy papers, and connections to other topics**
- **Demonstration: Gin**
- **Summary and Q&A**

Genetic Improvement of Software

human writes code

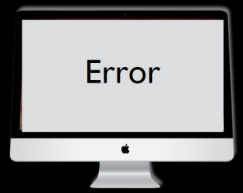
computer improves it

Justyna Petke

Functional Properties

LOGICAL

New Feature



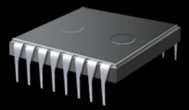
Bug Repair

accuracy

Non-Functional Properties

PHYSICAL

Execution Time



Memory

UNITS

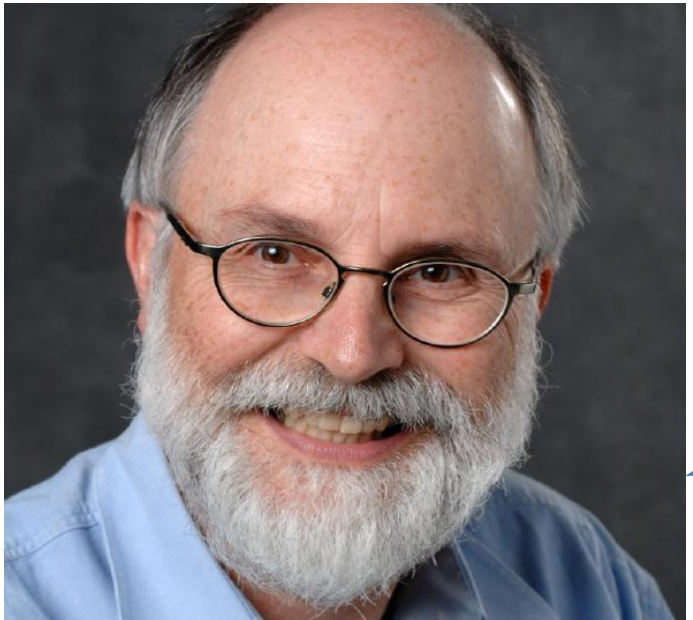
Bandwidth

Battery

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

Size

What is Genetic Improvement

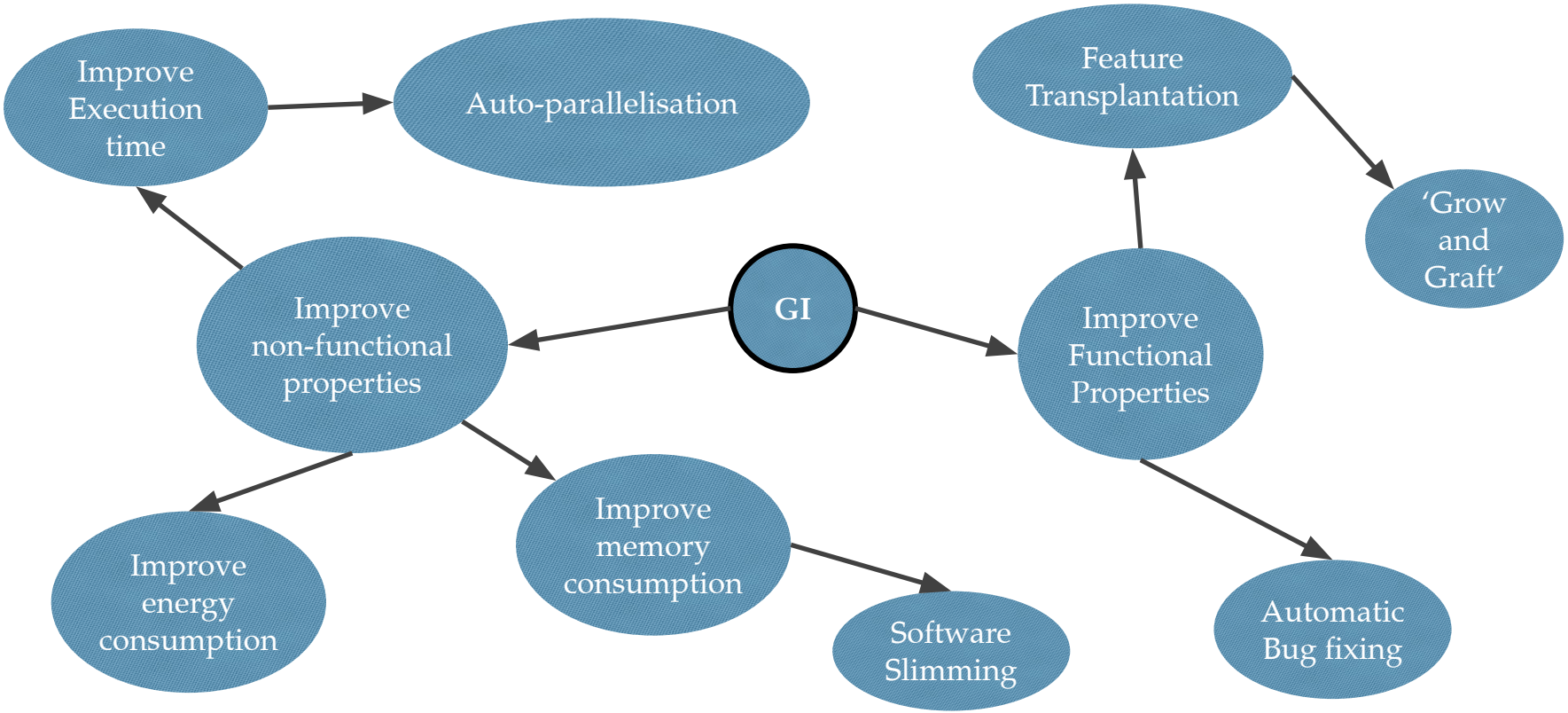


A wordy definition:

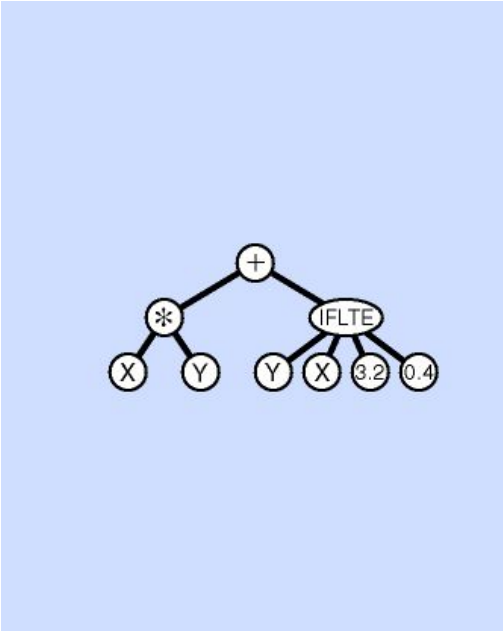
Genetic Improvement is the application of search-based (typically evolutionary) techniques to modify software with respect to some user-defined fitness measure.

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

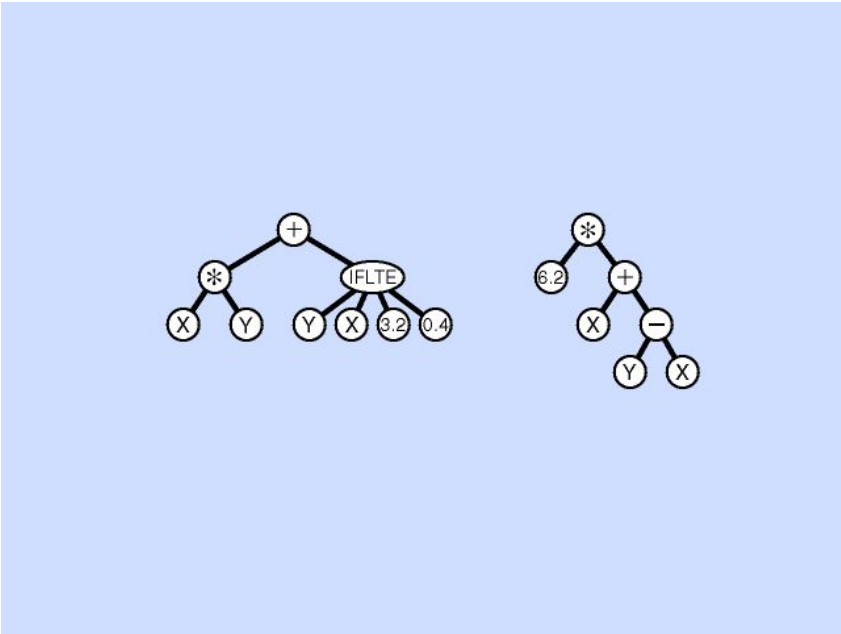
What is Genetic Improvement



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

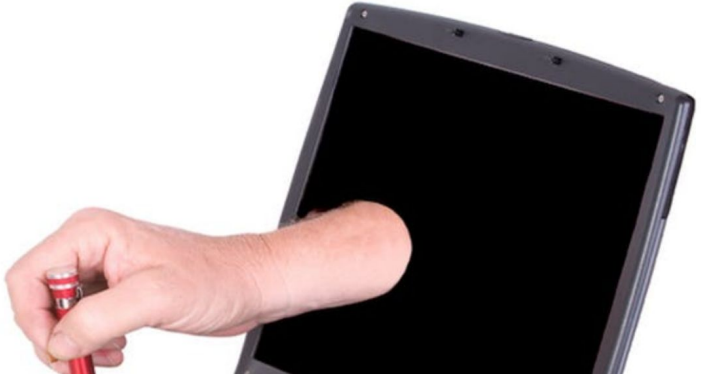
- easy to digest articles for non-specialists.

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?

IT?

October 12, 2017 3.29pm BST



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

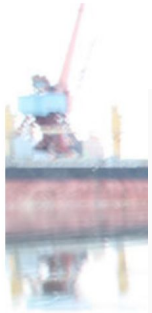
http://www.davidrwhite.co.uk/2014/11/27/genetic-programming-has-gone-backwards/

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/



Google

has genetic programming gone backwards

All Videos Images News Shopping More Sett

About 2,440,000 results (0.46 seconds)

TAG ARCHIVES: GEN

Genetic Programming has gone Backwards | David R. White

www.davidrwhite.co.uk/2014/11/27/genetic-programming-has-gone-backwards/

Genetic Improvement: the Story so far

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

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 (write 7 lines code, or utter 7 words **before they're 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]

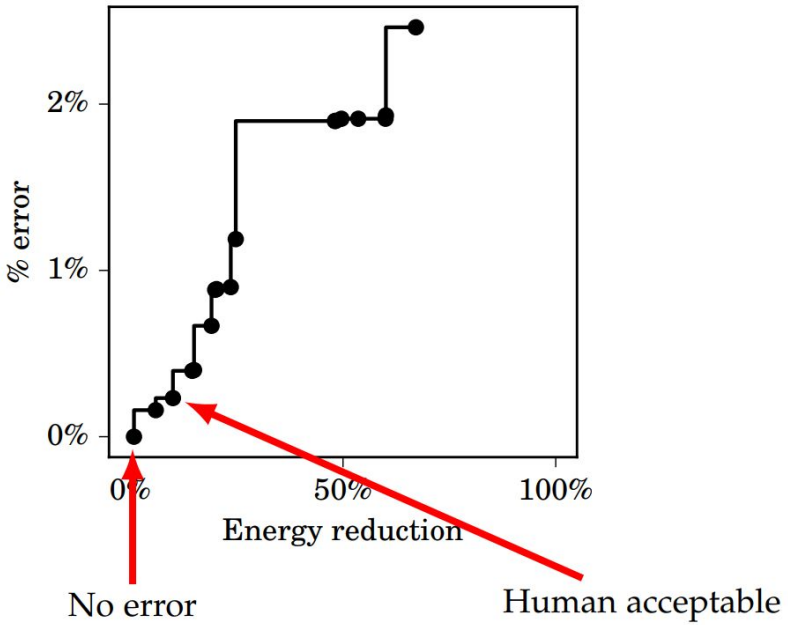
4. Errors, compile, halting (Langdon - discard)

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 has 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.

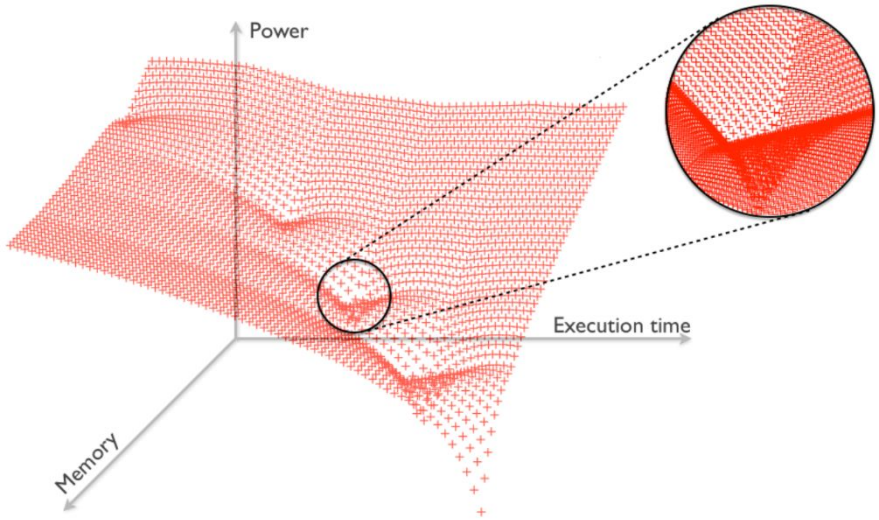


Figure 1: The GISMOE Pareto Program Surface

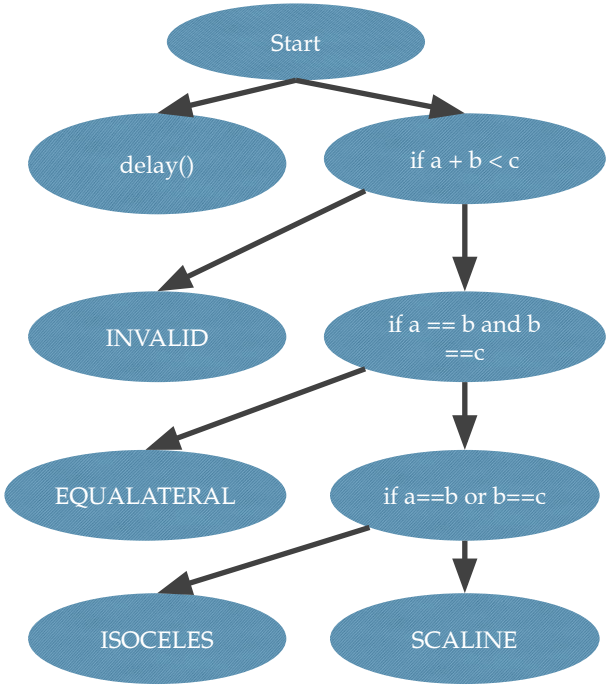
EDIT Operators – changes to programs

- 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.

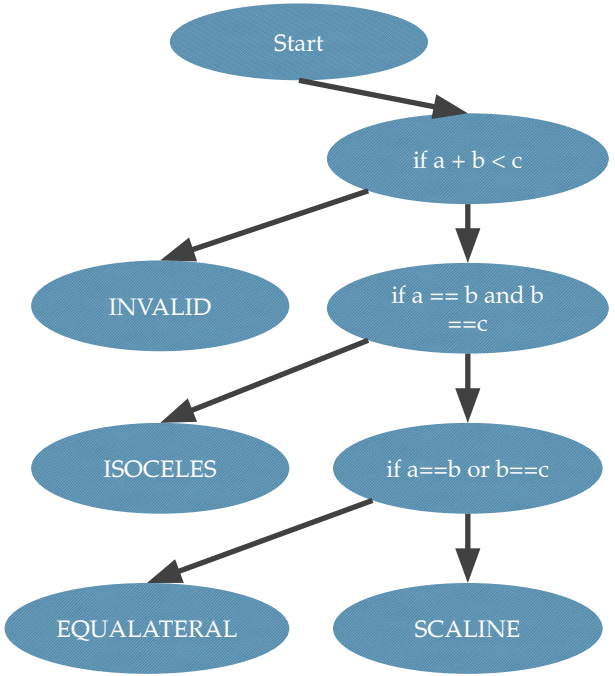
GI: An example of execution time optimisation

```
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
    }
}
```

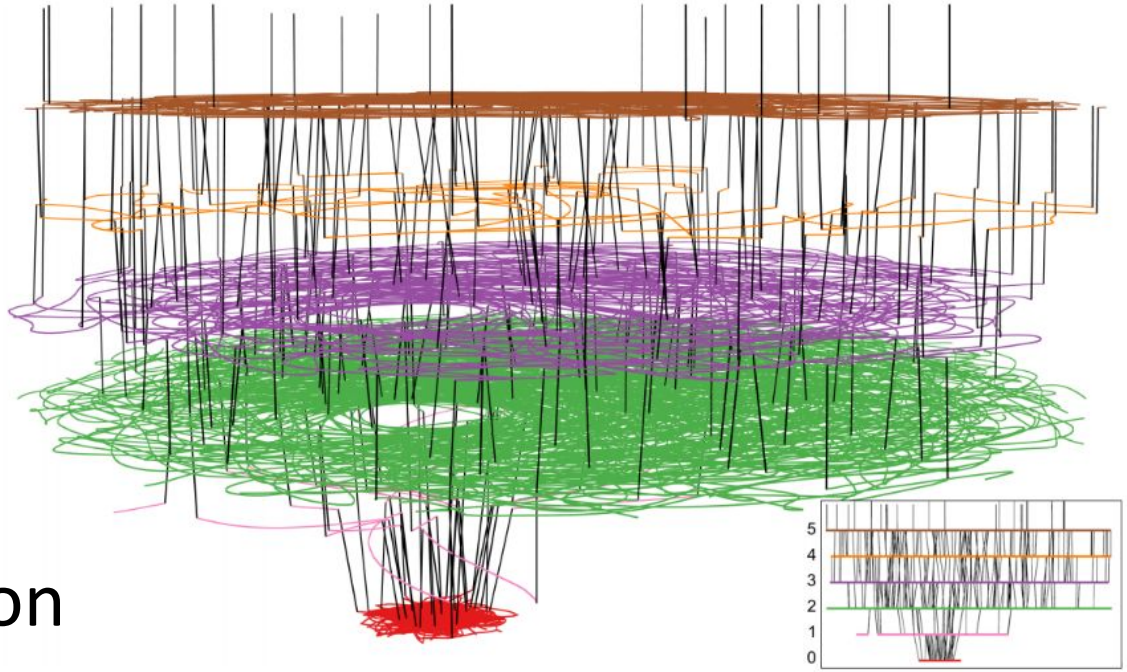


GI: An example of automated bug fixing

```
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
    }
}
```



Neutral networks Graceful degradation

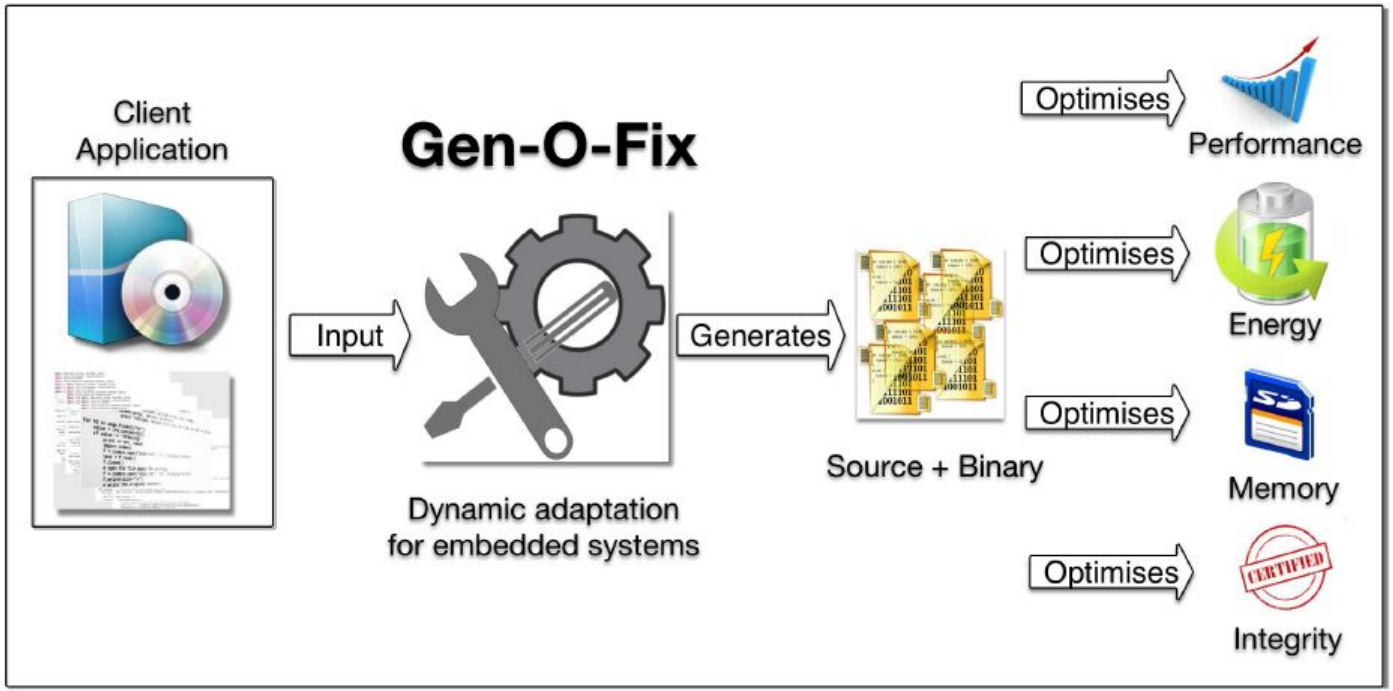


structure

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

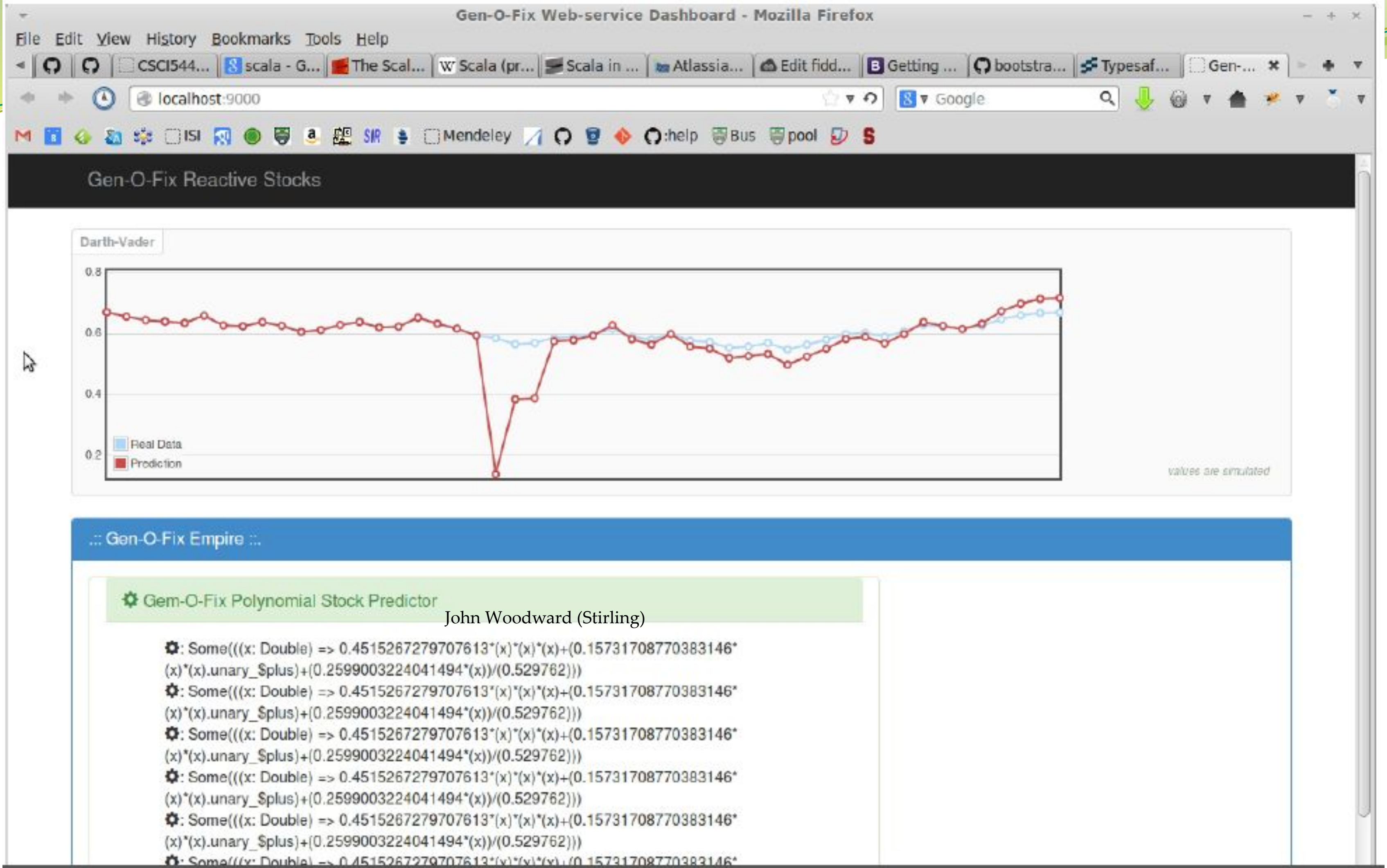
1. **Embedded** adaptively.
2. Minimal end-user requirements.
    1. Initial source code: **location** of Scala source code file containing a function
    2. Fitness function: providing a means of **evaluating the quality** of system
3. **Source to source transformations**
4. Operates on **ASTs** (i.e. 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
```


Gen-O-Fix Web-service Dashboard - Mozilla Firefox
File Edit View History Bookmarks Tools Help
localhost:9000
Gen-O-Fix Reactive Stocks
Darth-Vader
0.8
0.6
0.4
0.2
Real Data
Prediction
values are simulated
.:: Gen-O-Fix Empire ::.
Gem-O-Fix Polynomial Stock Predictor
: Some(((x: Double) => 0.4515267279707613*(x)*(x)*(x)+(0.1573170877038314 6*(x)*(x).unary_$plus)+(0.259900322404149 4*(x))/(0.529762)))
: Some(((x: Double) => 0.4515267279707613*(x)*(x)*(x)+(0.1573170877038314 6*(x)*(x).unary_$plus)+(0.259900322404149 4*(x))/(0.529762)))
: Some(((x: Double) => 0.4515267279707613*(x)*(x)*(x)+(0.1573170877038314 6*(x)*(x).unary_$plus)+(0.259900322404149 4*(x))/(0.529762)))
: Some(((x: Double) => 0.4515267279707613*(x)*(x)*(x)+(0.1573170877038314 6*(x)*(x).unary_$plus)+(0.259900322404149 4*(x))/(0.529762)))
: Some(((x: Double) => 0.4515267279707613*(x)*(x)*(x)+(0.1573170877038314 6*(x)*(x).unary_$plus)+(0.259900322404149 4*(x))/(0.529762)))

GI Hashcode tuning

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 of hashes (uniform distribution is the ideal), over 10,000 instances.

Distribution of Hashcodes

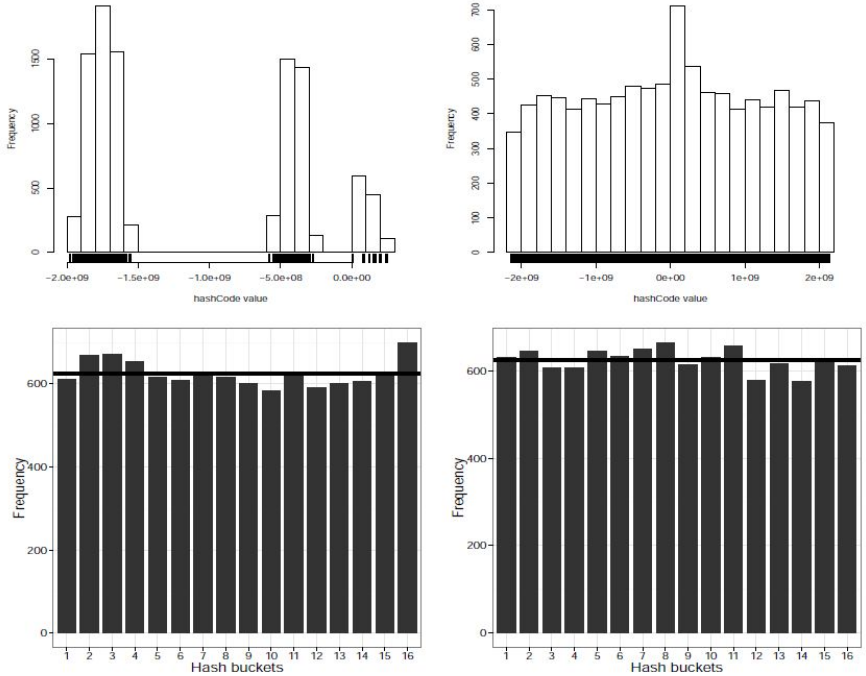


Fig. 1: The distribution of the hashcode values (top) and the distribution of the created objects in hash buckets (bottom), generated by the Apache commons (left) and the evolved function (right)

Overview

- **Introduction: why GI? And basic principles**
- **Challenges and open research questions**
- **Case study: fixing bugs**
- **The human perspective**
- **Noteworthy papers, and connections to other topics**
- **Demonstration: Gin**
- **Summary and Q&A**

Theory

- Hard!

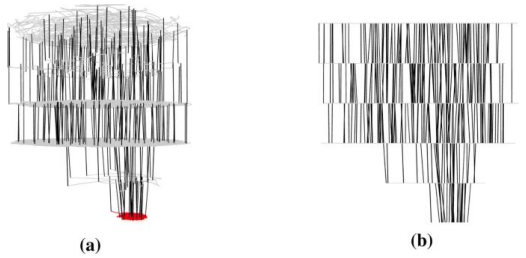
- NFL not really valid for GP, and therefore GI.
  - Why – because many programs share same functionality… and the NFL would assume that all program are equally likely (which is not the case in practical applications)

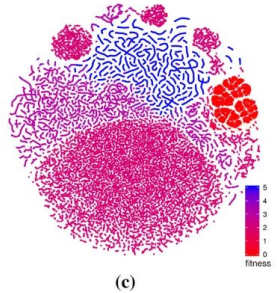
=> GI will remain empirical for years to come

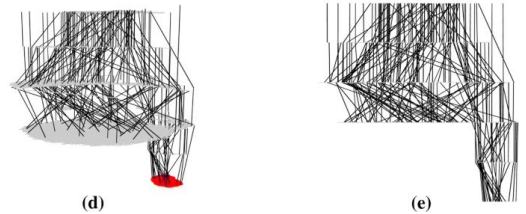
Theory

E.g. Landscapes…
Lots of neutrality!

Veerapen N, Ochoa G. Visualising the global structure of search landscapes: genetic improvement as a case study. Genetic programming and evolvable machines. 2018. 19(3):317-49





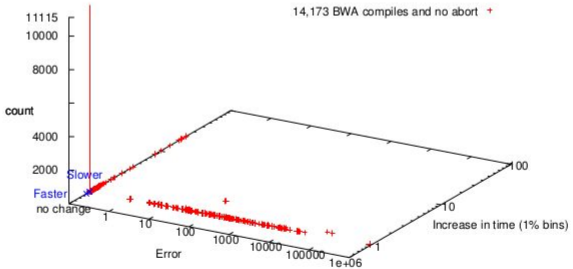


Software is Not Fragile

William B. Langdon and Justyna Petke

CREST, Department of Computer Science,
University College London Gower Street, London WC1E 6BT, UK

Abstract. Trying all simple changes (first order mutations) to executed C, C++ and CUDA source code shows software engineering artefacts are more robust than is often assumed. Of those that compile, up to 89% run without error. Indeed a few mutants are improvements. Program fitness landscapes are smoother. Analysis of these programs, a parallel nVidia GPGPU kernel, all CUDA samples and the GNU C library shows many lines of code and integer values are repeated and may follow Zipf's law.



Theory

E.g. Sampling of the space

Genetic Programming and Evolvable Machines (2019) 20:531–580
https://doi.org/10.1007/s10710-019-09355-3

Check for updates

A journey among Java neutral program variants

Nicolas Harrand[1] · Simon Allier[2] · Marcelino Rodriguez-Cancio[3] · Martin Monperrus[1] · Benoit Baudry[1]

Received: 18 December 2018 / Revi

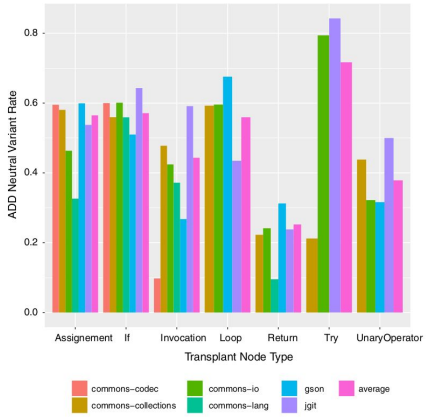


Fig. 6 Neutral variant rate for the add transformation, depending on the type of AST node used as trans-

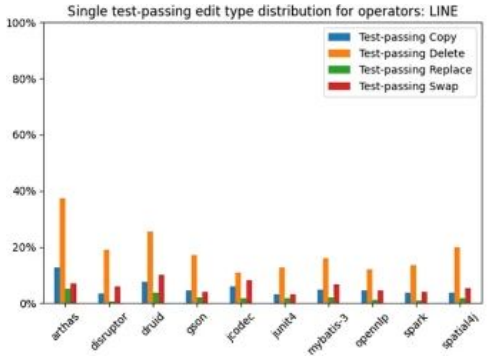
Empirical Software Engineering (2023) 28:104
https://doi.org/10.1007/s10664-023-10344-5

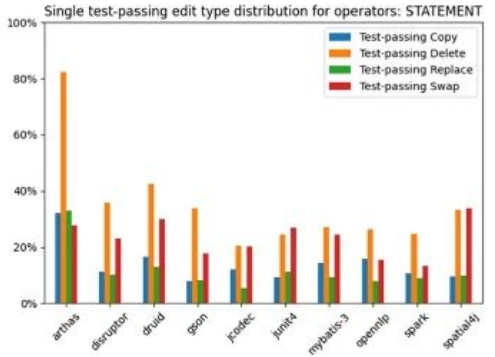
Check for updates

Program transformation landscapes for automated program modification using Gin

Justyna Petke[1] · Brad Alexander[2] · Earl T. Barr[1] · Alexander E.I. Brownlee[3] · Markus Wagner[4] · David R. White[5]

Accepted: 23 May 2023 / Published online: 14 July 2023





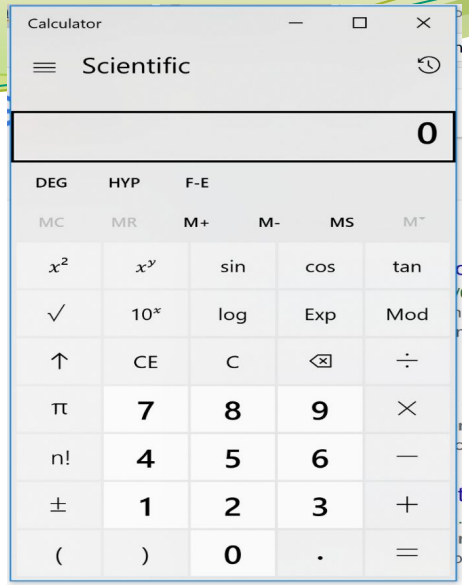
Theory

Lots remains!

Where and when does GI work best?

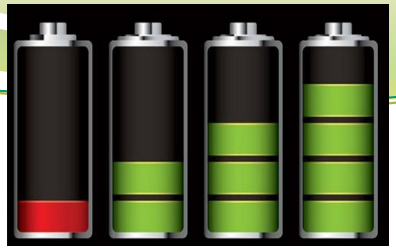
How does this vary for functionality / run time / energy … ?

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. Blot, Aymeric, and Justyna Petke. "A Comprehensive Survey of Benchmarks for Automated Improvement of Software's Non-Functional Properties" arXiv:2212.08540 (2022).
8. Benchmarking is more complex (noise, hardware, prog lang, …)

Measuring Energy



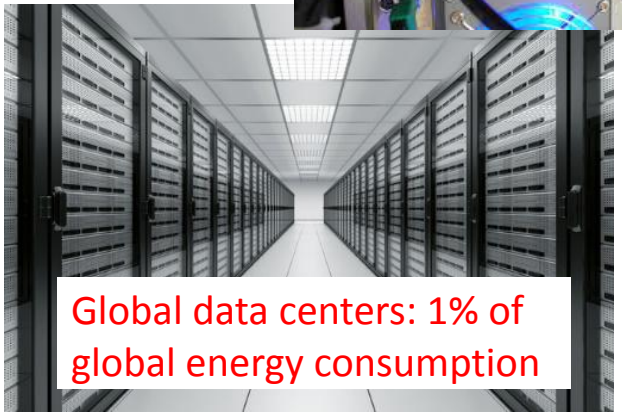
- **computational energy consumption growing importance, particularly at the extremes (i.e., mobile devices and datacentres).**

one line = one unit

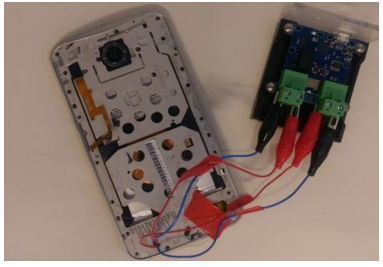
simulate (runtime/system calls/) Tools Opacitor, PowerGauge

read battery indicator

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



Global data centers: 1% of global energy consumption



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]

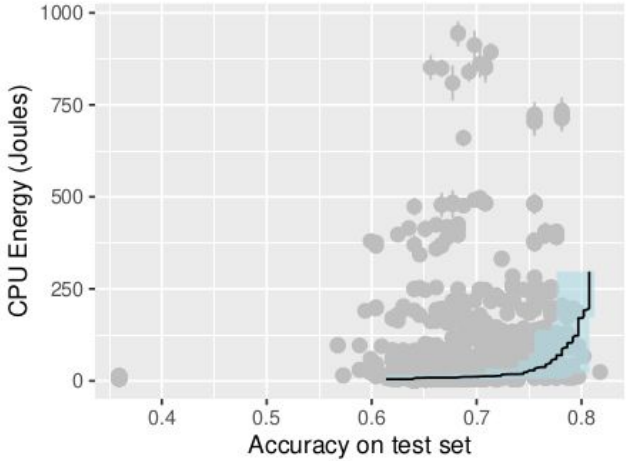
Measuring Energy

Trade-offs to exploit, but lots of noise and many confounding factors

Exploring the Accuracy – Energy Trade-off in Machine Learning

Alexander E.I Brownlee, Jason Adair, Saemundur O. Haraldsson and John Jabbo

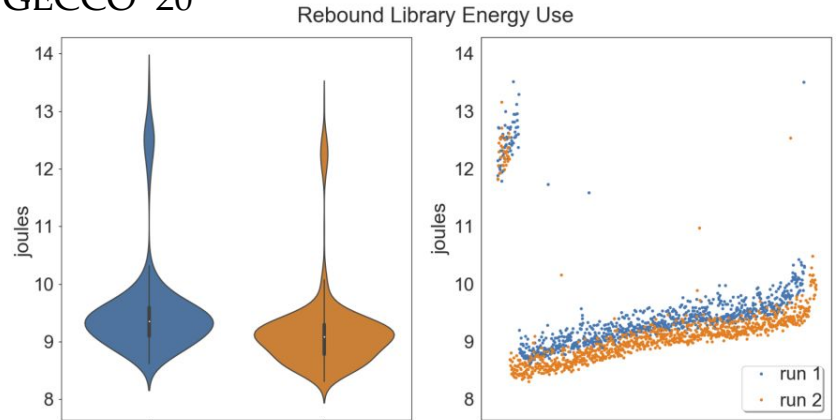
GI@ICSE'21



Towards Rigorous Validation of Energy Optimisation Experiments

Mahmoud A. Bokhari Brad Alexander, Markus Wagner

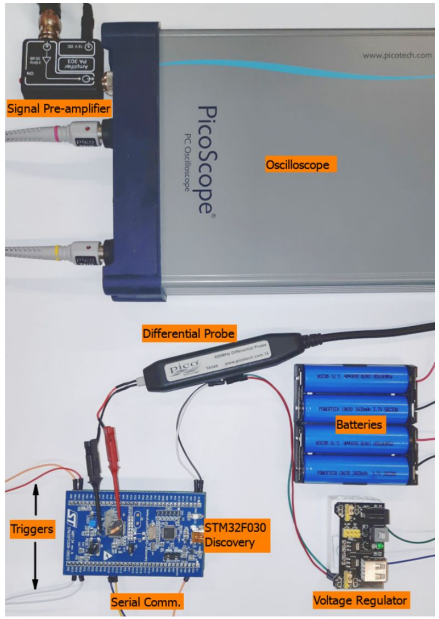
GECCO '20



Shown: measurements of the same task… just the phone was recharged and restarted between run1 and run2

GI to eliminate side-channels

Minimise the “signal” when performing cryptographic operations, as they can leak information on the secret key via power consumption profiling!



CCS’22

ROSITA: Towards Automatic Elimination of Power-Analysis Leakage in Ciphers

Madura A. Shelton
University of Adelaide
madura.shelton@adelaide.edu.au

Niels Samwel
Radboud University
nsamwel@cs.ru.nl

Lejla Batina
Radboud University
lejla@cs.ru.nl

Francesco Regazzoni
University of Amsterdam and ALaRI – USI
f.regazzoni@uva.nl, regazzoni@alari.ch

Markus Wagner
University of Adelaide
markus.wagner@adelaide.edu.au

Yuval Yarom
University of Adelaide and Data61
yval@cs.adelaide.edu.au

NDSS’21

ROSITA++: Automatic Higher-Order Leakage Elimination from Cryptographic Code

Madura A. Shelton
University of Adelaide
Australia
madura.shelton@adelaide.edu.au

Łukasz Chmielewski
Radboud University and Riscure
The Netherlands
lukasz@cs.ru.nl

Niels Samwel
Radboud University
The Netherlands
nsamwel@cs.ru.nl

Markus Wagner
University of Adelaide
Australia
markus.wagner@adelaide.edu.au

Lejla Batina
Radboud University
The Netherlands
lejla@cs.ru.nl

Yuval Yarom
University of Adelaide
Australia
yval@cs.adelaide.edu.au

GI to improve Speed

Also noisy… also platform dependent (read: you can specialise code to architectures)

PLDI'23
Distinguished Paper
Humies'23 Gold Award

Key aspects:

- Demonstrated on straight line code (cryptographic primitives)
- Optimisation at the levels of: intermediate representation & Assembly
- *Automatic formal proofs of correctness*
- Yields new, fastest implementations
- Curve25519 Code in BoringSSL
  ⇒ this runs in your Chrome and Edge browser now!

CryptOpt: Verified Compilation with Randomized Program Search for Cryptographic Primitives

JOEL KUEPPER, University of Adelaide, Australia
ANDRES ERBSEN, Massachusetts Institute of Technology, USA
JASON GROSS, Massachusetts Institute of Technology, USA
OWEN CONOLY, Massachusetts Institute of Technology, USA
CHUYUE SUN, Stanford University, USA
SAMUEL TIAN, Massachusetts Institute of Technology, USA
DAVID WU, University of Adelaide, Australia
ADAM CHLIPALA, Massachusetts Institute of Technology, USA
CHITCHANOK CHUENGSATIANSUP, The University of Melbourne, Australia
DANIEL GENKIN, Georgia Institute of Technology, USA
MARKUS WAGNER, Monash University, Australia
YUVAL YAROM, Ruhr University Bochum, Germany

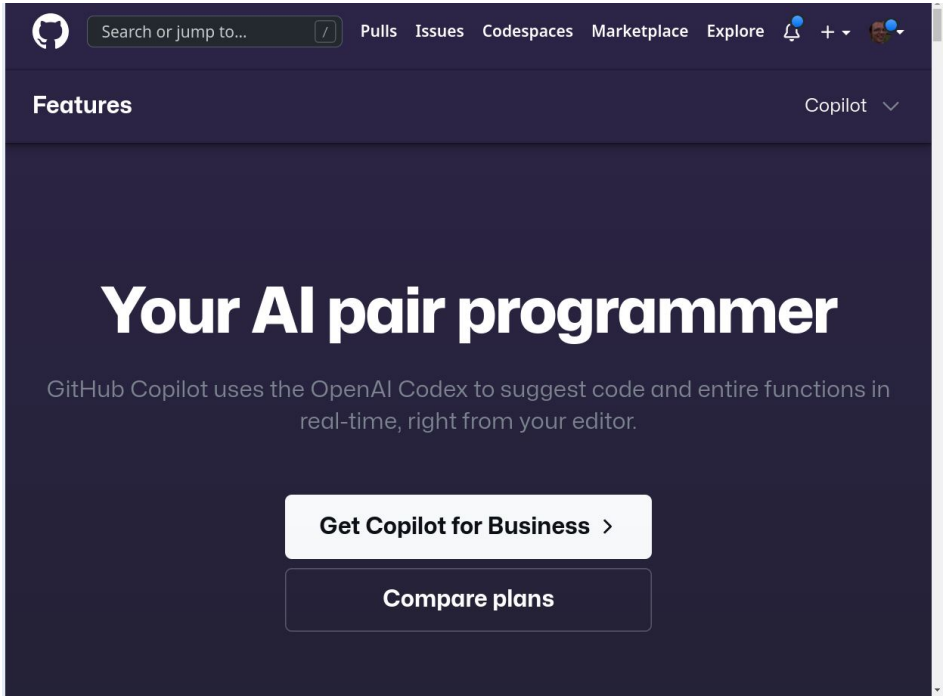
What about Copilot/ChatGPT…?

Large language models generate code!

Replicate patterns given some prompt

Can lead to errors![*] Related-but-incorrect solutions

GI search tests the code as it goes, so can be constrained to only produce variants that (probably) work



*Jones E & Steinhardt J. Capturing failures of large language models via human cognitive biases. In AH Oh, A Agarwal, D Belgrave & K Cho, eds., Advances in Neural Information Processing Systems. 2022

Towards Objective-Tailored Genetic Improvement Through Large Language Models

Sungmin Kang
KAIST
sungmin.kang@kaist.ac.kr

Shin Yoo
KAIST
shin.yoo@kaist.ac.kr

12th International Workshop on Genetic Improvement
ICSE 2023

Enhancing Genetic Improvement Mutations Using Large Language Models

Alexander E.I. Brownlee[1 [0000−0003−2892−5059]], James Callan[2 [0000−0002−5692−6203]], Karine Even-Mendoza[3 [0000−0002−3099−1189]], Alina Geiger[4 [0009−0002−3413−283X]], Carol Hanna[2 [0009−0009−7386−1622]], Justyna Petke[2 [0000−0002−7833−6044]], Federica Sarro[2 [0000−0002−9146−442X]], and Dominik Sobania[4 [0000−0001−8873−7143]]

[1] University of Stirling, UK
[2] University College London, UK
[3] King's College London, UK
[4] Johannes Gutenberg University Mainz, Germany

SSBSE Challenge Track 2023

An Analysis of the Automatic Bug Fixing Performance of ChatGPT

Dominik Sobania
Johannes Gutenberg University Mainz
Email: dsobania@uni-mainz.de

Martin Briesch
Johannes Gutenberg University Mainz
Email: briesch@uni-mainz.de

Carol Hanna
University College London
Email: carol.hanna.21@ucl.ac.uk

Justyna Petke
University College London
Email: j.petke@ucl.ac.uk

2023 IEEE/ACM International Workshop on Automated Program Repair (APR), 2023

Creative and Correct: Requesting Diverse Code Solutions from AI Foundation Models

Scott Blyth
sbly0002@student.moansh.edu
Monash University
Clayton, Australia

Markus Wagner
markus.wagner@monash.edu
Monash University
Clayton, Australia

Christoph Treude
ctreude@smu.edu.sg
Singapore Management University
Singapore, Singapore

FORGE 2024

Overview

- **Introduction: why GI? And basic principles**
- **Challenges and open research questions**
- **Case study: fixing bugs**
- **The human perspective**
- **Noteworthy papers, and connections to other topics**
- **Demonstration: Gin**
- **Summary and Q&A**

Case study

Fixing bugs: A real world example of GI in action

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

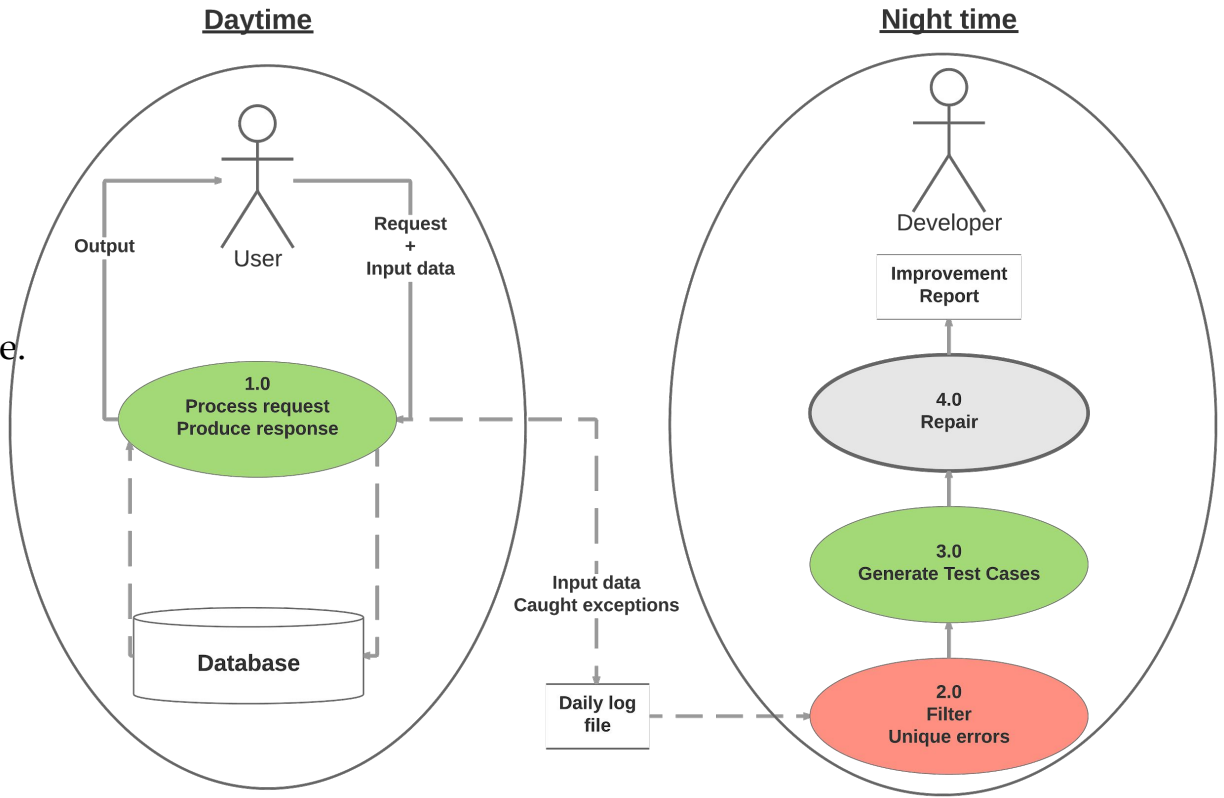
S.O. Haraldsson, John R. Woodward, Alexander E. I. Brownlee, Albert V. Smith, and Vilmundur Gudnason. 2017. Genetic improvement of runtime and its fitness landscape in a bioinformatics application. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '17). ACM, New York, 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

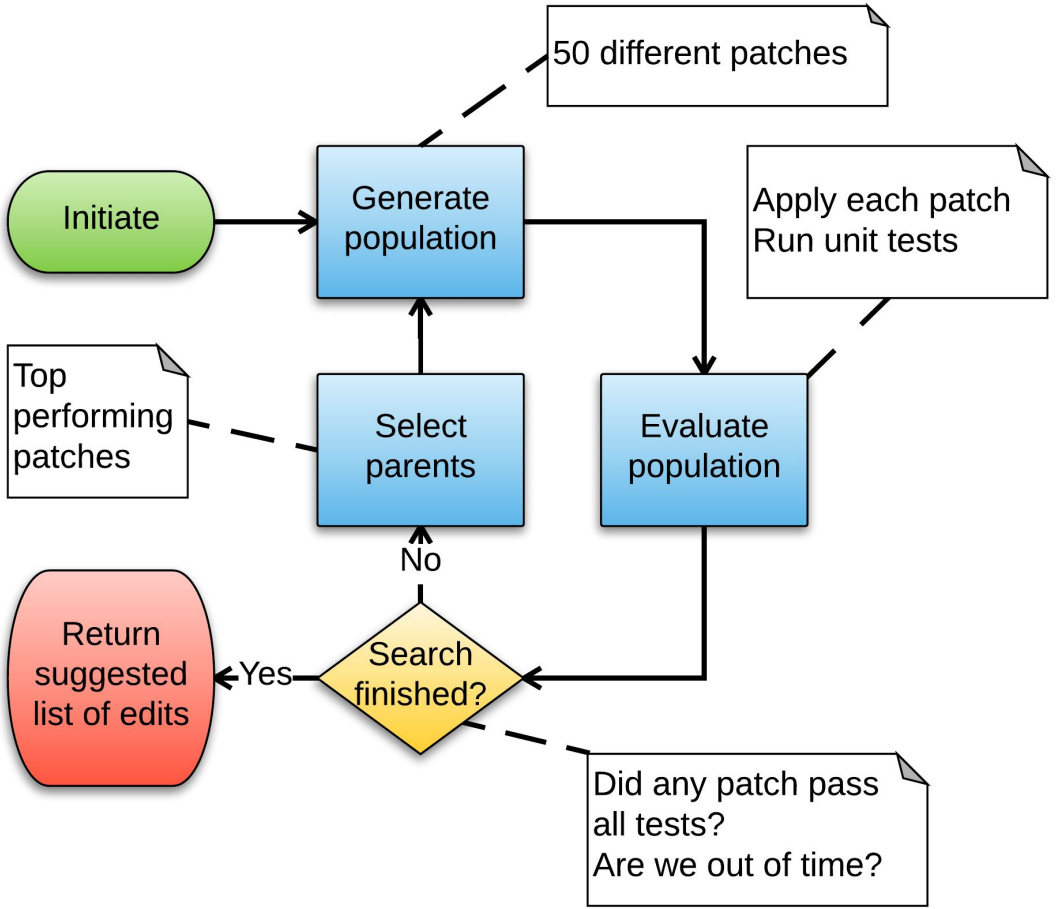
G. B. Saemundsdottir, and S.O. Haraldsson 2024. "Large Language Models as All-in-One Operators for Genetic Improvement.", to appear in Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '24). ACM, New York, NY, USA, 1501-1508. URL: https://doi.org/10.1145/3638530.3654408

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



- Procedure 4.0
- Genetic Improvement
  - Pop.= 50 patches
  - fit.= #passed tests
  - select= ½ pop by fitness
  - Output= report



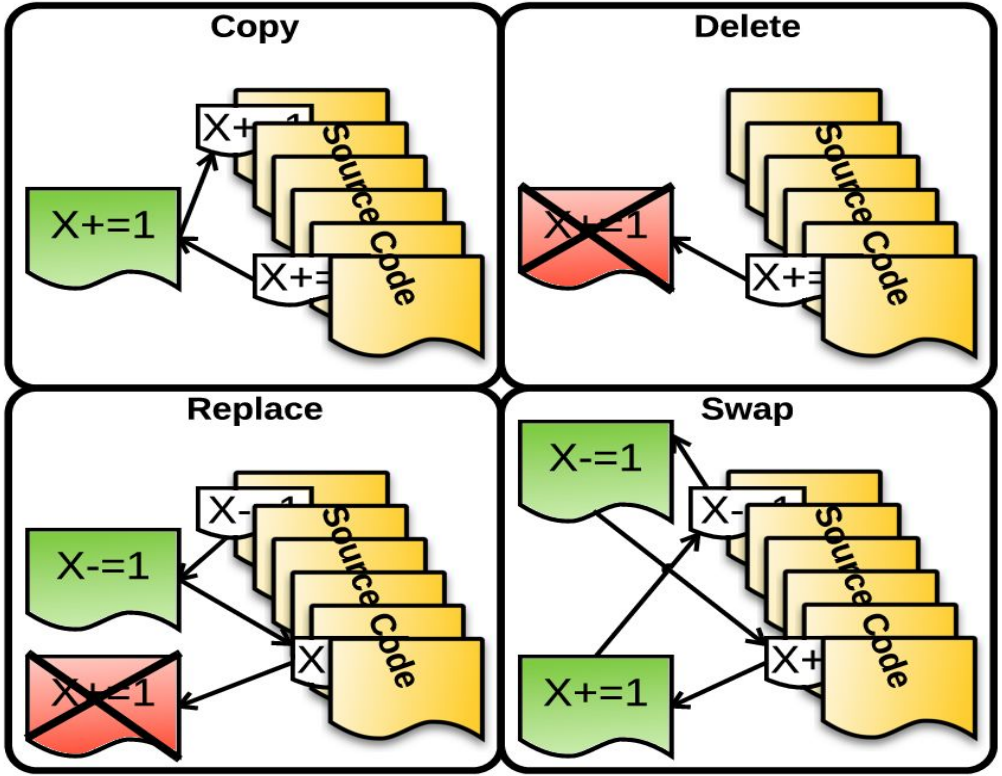
4 different types of implemented **Edits**

Primitive types:

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

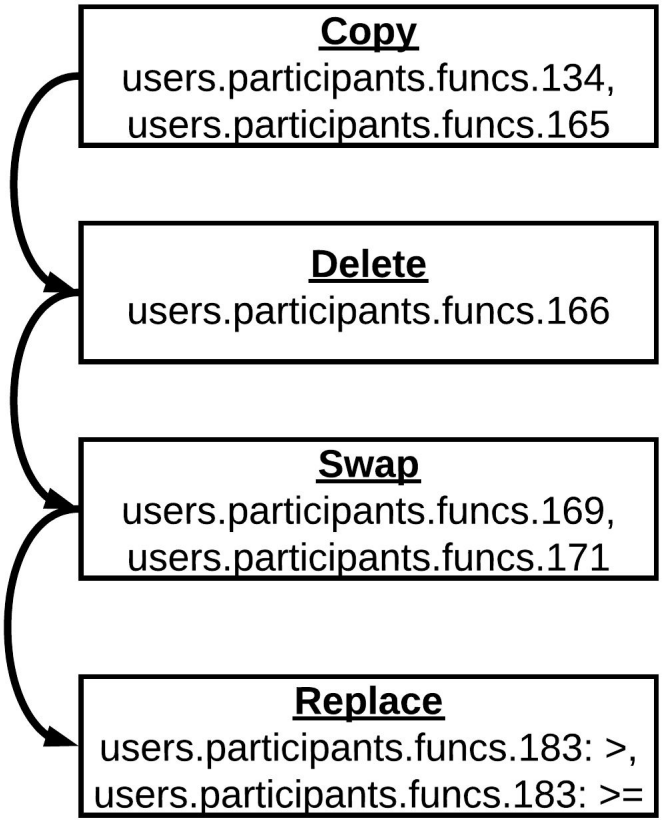
Composite types:

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



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



Or just let an LLM do the work

Points to keep in mind though:

Representation
- **Of the Code**
- **Of the Changes**

Reliability
- **Of the Prompts**
- **Of the User**

Readability
- **To the human**

Efficiency vs Workload distribution
- **Prompt engineering**
  - **vs**
- **Computations (The GI search)**
  - **vs**
- **The LLM Model**

Overview

- **Introduction: why GI? And basic principles**
- **Challenges and open research questions**
- **Case study: fixing bugs**
- **The human perspective**
- **Noteworthy papers, and connections to other topics**
- **Demonstration: Gin**
- **Summary and Q&A**

The human perspective

E. R. Winter et al., "Let's Talk With Developers, Not About Developers: A Review of Automatic Program Repair Research," in IEEE Transactions on Software Engineering, doi: https://doi.org/10.1109/TSE.2022.3152089

V. Nowack et al., "Expanding Fix Patterns to Enable Automatic Program Repair," 2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE), 2021, pp. 12-23, doi: https://10.1109/ISSRE52982.2021.00015 .

E. Winter et al., "How do Developers Really Feel About Bug Fixing? Directions for Automatic Program Repair," in IEEE Transactions on Software Engineering, vol. 49, no. 4, pp. 1823-1841, 1 April 2023, doi: 10.1109/TSE.2022.3194188.

For the researcher

For the developer

Overview

- **Introduction: why GI? And basic principles**
- **Challenges and open research questions**
- **Case study: fixing bugs**
- **The human perspective**
- **Noteworthy papers, and connections to other topics**
- **Demonstration: Gin**
- **Summary and Q&A**

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.

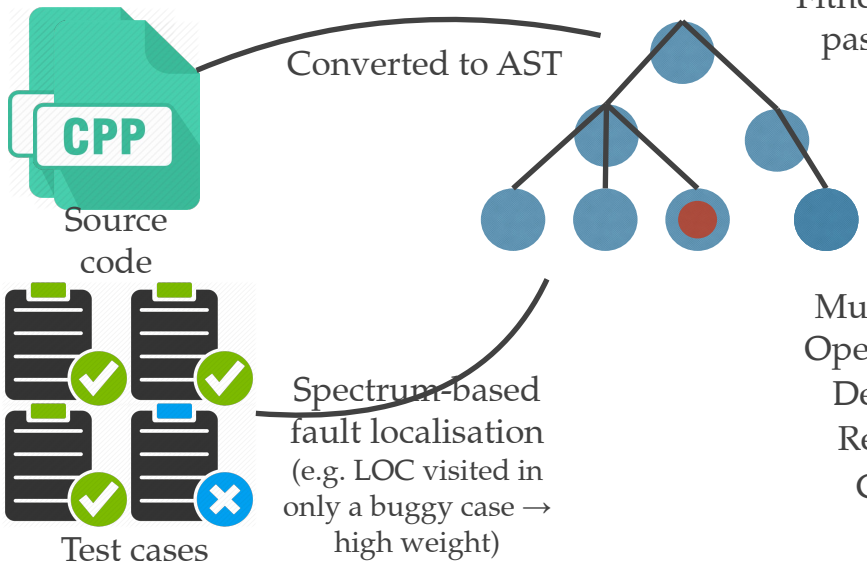
A Systematic Study of Automated Program Repair: Fixing 55 out of 105 Bugs for $8 Each

(2012)
Cited >700 times

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

Stephanie Forrest
Computer Science Department
University of New Mexico
Albuquerque, NM
forrest@cs.unm.edu

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



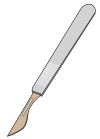
- Where an adequate test suite is provided, GenProg has been shown to fix **real-world bugs**
- It has **inspired a variety of alternative frameworks,** most of which claim to outperform GenProg

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.13,j.petke}@ucl.ac.uk

Featured in:



muScalpel

Donor

Host

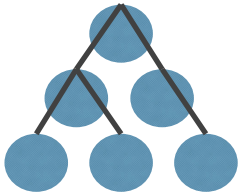
(2014)

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

Mark Harman, Yue Jia, and William B. Langdon

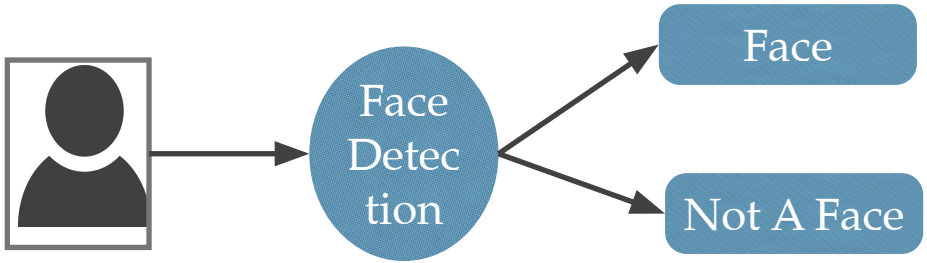
University College London, CREST centre, UK

English to Korean;
English to Portuguese



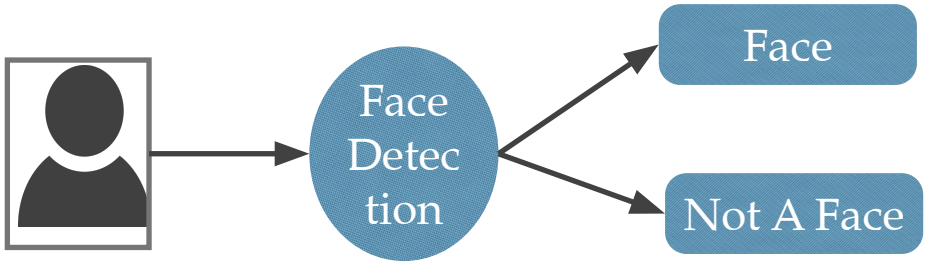
Deep Parameter Optimisation for Face Detection Using the Viola-Jones Algorithm in OpenCV

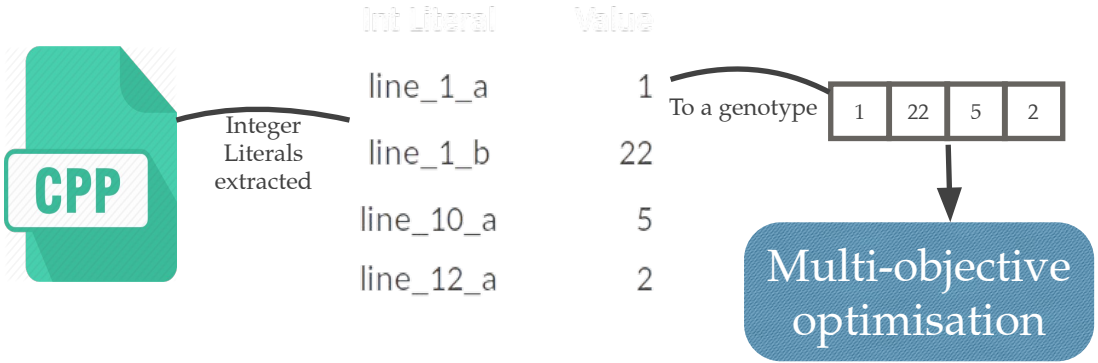
Bobby R. Bruce[1(✉)], Jonathan M. Aitken[2(✉)], and Justyna Petke[1(✉)]



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

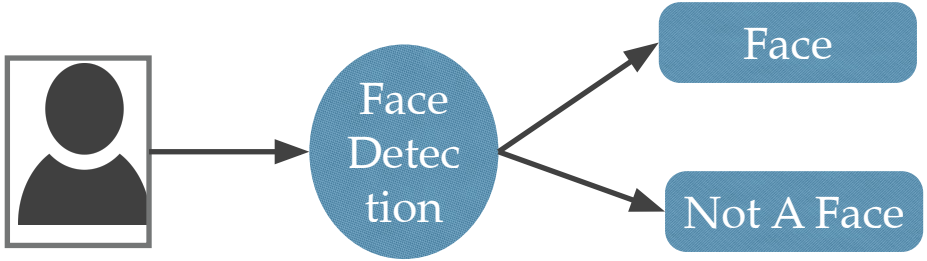
Bobby R. Bruce[1(✉)], Jonathan M. Aitken[2(✉)], and Justyna Petke[1(✉)]

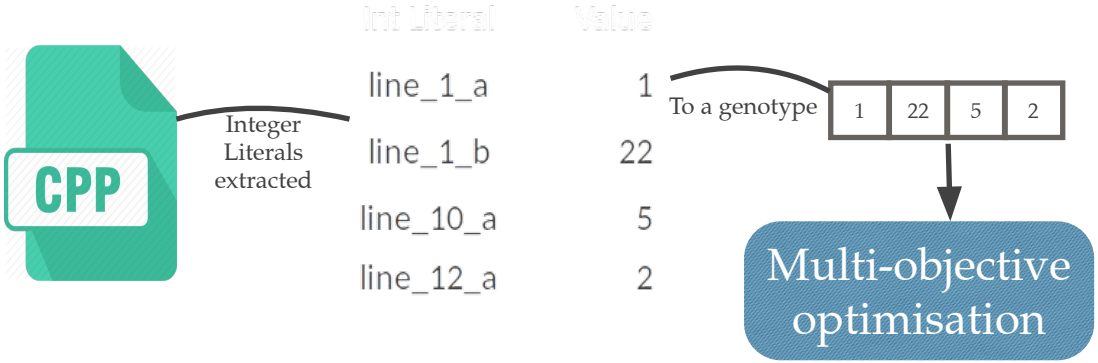




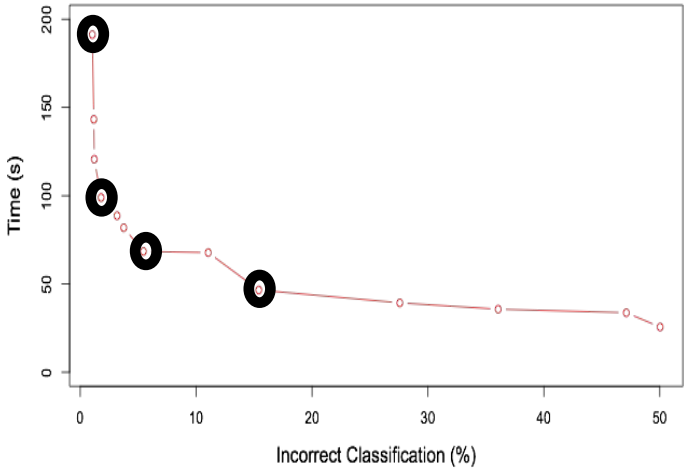
Deep Parameter Optimisation for Face Detection Using the Viola-Jones Algorithm in OpenCV

Bobby R. Bruce[1(✉)], Jonathan M. Aitken[2(✉)], and Justyna Petke[1(✉)]





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



Genetic Improvement of Data gives double precision invsqrt

W. B. Langdon
Department of Computer Science, University College London Gower Street, WC1E 6BT, UK
w.langdon@cs.ucl.ac.uk

ABSTRACT

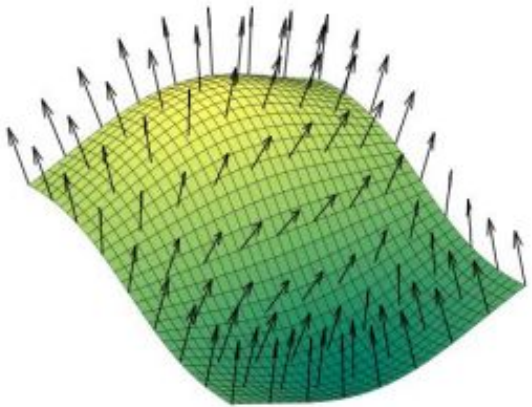
CMA-ES plus manual code changes rapidly transforms 512 Newton-Raphson start points from a GNU C library table driven version of sqrt into a double precision reciprocal square root function. The GI $x^{-\frac{1}{2}}$ is far more accurate than Quake's InvSqrt, Quare root.

CCS CONCEPTS

• **Software and its engineering → Search-based software engineering**;

KEYWORDS

evolutionary computing, Evolution Strategies, GGGP, search based software engineering, SBSE, software maintenance of literals, data transplantation, glibc, Newton's method

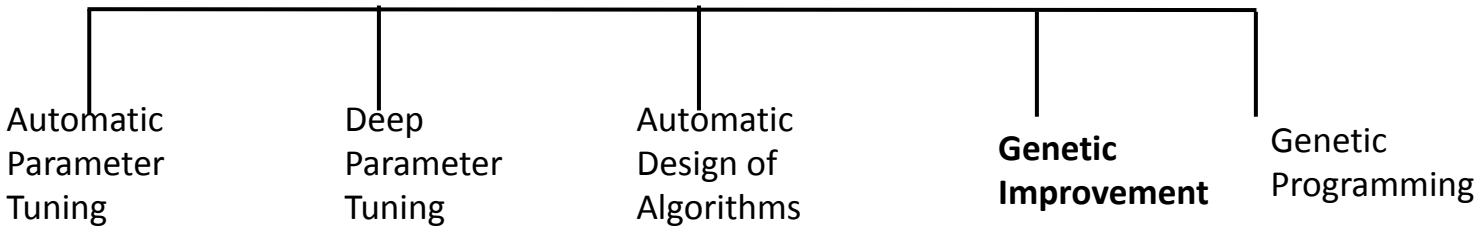


Phd Theses

- David R. White
- Andrea Arcuri
- Bobby R. Bruce
- Sæmundur Ó. Haraldsson
- Mahmoud R. Bokhari
- Michail Basios
- 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



Starting point – Surveys

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

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

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

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

Starting point – Surveys

Large Language Models for Software Engineering: Survey and Open Problems

arXiv
2023

Angela Fan
Generative AI Team
Meta Platforms Inc.
New York, NY, USA

Beliz Gokkaya
PyTorch Team
Meta Platforms Inc.
Menlo Park, CA, USA

Mark Harman
Instagram Product Foundation
Meta Platforms Inc.
London, UK

Mitya Lyubarskiy
Developer Infrastructure
Meta Platforms Inc.
London, UK

Shubho Sengupta
FAIR
Meta Platforms Inc.
Menlo Park, CA, USA

Shin Yoo
School of Computing
KAIST
Daejeon, Korea

Jie M. Zhang
Department of Informatics
King's College London
London, UK

A Comprehensive Survey of Benchmarks for Automated Improvement of Software's Non-Functional Properties

arXiv
2022

Starting point – Websites

- http://geneticimprovementofsoftware.com/
- https://geneticimprovementofsoftware.com/learn/survey - living survey
- https://en.wikipedia.org/wiki/Genetic_improvement_(computer_science)

Overview

- **Introduction: why GI? And basic principles**
- **Challenges and open research questions**
- **Case study: fixing bugs**
- **The human perspective**
- **Noteworthy papers, and connections to other topics**
- **Demonstration: Gin**
- **Summary and Q&A**

Demonstration: Gin

Gin

Genetic Improvement in No time

Toolbox for GI research targeting Java

https://github.com/gintool/gin

Initiated by David White, developed through community effort

Other tools exist! For example:

GI: PyGGI, locoGP

APR: ASTOR, GenProg

See also: Zuo, Blot, Petke: *Evaluation of genetic improvement tools for improvement of non-functional properties of software*. GECCO '22

Goals / principals

What's in Gin?
Pipelines
Example Edits
Sampling and Searching
Papers

Gin's Goals / Principals

A toolkit to enable experimentation

Remove *incidental* difficulties of GI for research and teaching

Work on open-source software projects out-of-the-box

Gin's Goals / Principals

Java

Maven

Gradle Build Tool

JAVAPARSER

FOR PROCESSING JAVA CODE

Apache Commons
http://commons.apache.org/

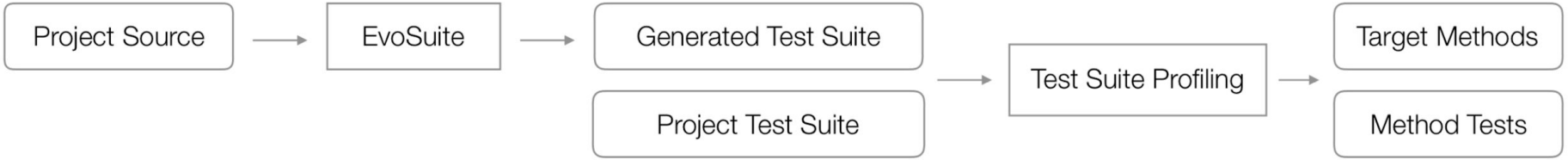
GUAVA

Libraries

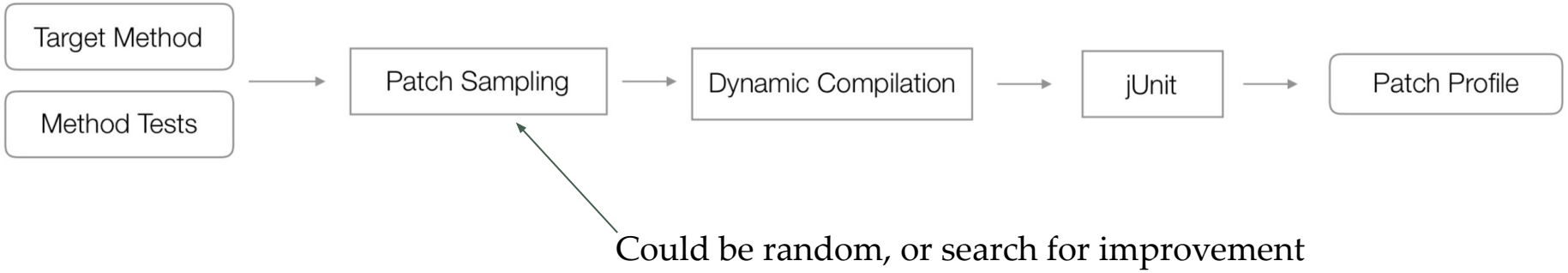
License

Gin Pipelines

Preprocessing



Search Space Analysis



Edits

- Edits are single changes to source code
  - Building blocks of a repair
  - Combined into Patches
- 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 ⇔AND) (++ ⇔ --)
    - reorder Boolean expressions (X && Y ⇔ Y && X)
    - loop and method shortcuts (insert return/break/continue)
  - LLM edits 

Edits

- Gin also provides tools to make designing your own edits 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 }
```

Disclaimer: this is a simplified version to illustrate. Some detail (e.g. serialisation, and code to avoid replacing statements within the same parent node) is omitted

Example edits

```
public InsertBreak(SourceFile sourceFile, Random rng) {
    SourceFileTree sf = (SourceFileTree) sourceFile;
    List<Integer> targetMethodBlocks = sf.getBlockIDsInTargetMethod();
    int insertBlock = targetMethodBlocks.get(rng.nextInt(targetMethodBlocks.size()));
    int insertStatementID = sf.getRandomInsertPointInBlock(insertBlock, rng);
    if (insertStatementID < 0) {
        insertStatementID = 0; // insert at start of empty block
    }
    this.destinationFilename = sourceFile.getRelativePathToWorkingDir();
    this.destinationBlock = insertBlock;
    this.destinationChildInBlock = insertStatementID;
}
```

...

```
public SourceFile apply(SourceFile sourceFile) {
    SourceFileTree sf = (SourceFileTree) sourceFile;
    BreakStmt toInsert = new BreakStmt();
    toInsert.removeLabel(); // a bit weird but if we don't do this we get "break empty;"
    // insertStatement will also just do nothing if the destination block is deleted
    sf = sf.insertStatement(destinationBlock, destinationChildInBlock, toInsert);
    return sf;
}
```

Patch / Edit Evaluation

Gin invokes test cases via Junit and tracks:

- compile success;
- run-time errors, exception types
- actual & expected outcomes
- timing: wall-clock and CPU time; peak memory; (energy coming soon)

```
UnitTest[] ut = {
    new UnitTest("TriangleTest", "testInvalidTriangles"),
    new UnitTest("TriangleTest", "testEqualateralTriangles"),
    new UnitTest("TriangleTest", "testIsocelesTriangles"),
    new UnitTest("TriangleTest", "testScaleneTriangles")
};

UnitTest.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 test0TimedOut = rs.getResults().get(0).getTimedOut();
long test0ExecutionTime = rs.getResults().get(0).getExecutionTime();
String test0ExceptionMessage = rs.getResults().get(0).getExceptionMessage();
```

Sampling and Searching

- Included samplers:
  - EmptyPatchTester
  - RandomSampler
  - DeleteEnumerator
- Searches: LocalSearch, GP, NSGA-II

- 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?
  - ...

DeleteEnumerator

```
1  public static void main(String[] args) {
2
3    UnitTest[] 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 }
```

Random Sampling Output

The following is one really wide output file…

PatchIndex	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/containers/mp4/boxes/Box.java:514 -> ./src/main/java/org/jcodec/containers/mp4/boxes/Box.java:110:110 |

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.containers.mp4.boxes.TrunBoxTest.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).
           totalExecutionTime();
5
6      for (int step = 1; step <= NUM_STEPS; step++) {
7          Patch neighbour = neighbour(bestPatch, rng);
8          UnitTestResultSet rs = testRunner.test(neighbour
               ,10);
9          if (rs.getValidPatch() && rs.getCleanCompile() &&
10             rs.allTestsSuccessful() &&
11             rs.totalExecutionTime() < bestTime) {
12           bestPatch = neighbour;
13           bestTime = rs.totalExecutionTime();
14         }
15     }
16
17     return bestPatch;
18 }
19
20 public Patch neighbour(Patch patch, Random rng) {
21     Patch neighbour = patch.clone();
22
23     if (neighbour.size() > 0 && rng.nextFloat() > 0.5) {
24         neighbour.remove(rng.nextInt(neighbour.size()));
25     } else {
26         neighbour.addRandomEdit(rng, allowableEditTypes);
27     }
28
29     return neighbour;
30 }
```

Local search, output

```
-bash-4.1$ java -jar build/gin.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: 1646971219ns
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:28Failed 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:35
|, 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: 1647522167ns
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: 1648831018ns
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:39 <-> examples/triangle/Triangle.java:29 |, New best time: 38744892(ns)
2020-04-10 04:36:29 gin.LocalSearch.search() INFO: Finished. Best time: 38744892 (ns), Speedup (%): 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 SCALENE = 1;
        static final int EQUALATERAL = 2;
        static final int ISOCELES = 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 ISOCELES;
        } else {
        return SCALENE;
        }

        }

        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 SCALENE = 1;
        static final int EQUALATERAL = 2;
        static final int ISOCELES = 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 ISOCELES;
        } else {
        return SCALENE;
        }

        }

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

        }
        }

}
```

LLM support

Integration with:

- OpenAI API via langchain4j
- ollama for many other local LLMs

Plain text templates for prompts

Possibility to pass error messages etc. in the prompt

Currently in a development branch “llm” - full integration coming soon

LLM support

simple-prompt.txt

```
Give me $COUNT$ different Java implementations of this method body:```
$DESTINATION$
```
This code belongs to project $PROJECT$.

Wrap all code in curly braces, if it is not already. Do not include any method or class declarations. Label all code as java.
```

```
> projectnameforgin='jcodec'
> sampler='RandomSampler'
> editCount='1000'
> modelType='OpenAI'
> java -Dtinylog.level=trace -cp ../gin-llm/build/gin.jar gin.util.$sampler -j -p $projectnameforgin -d . -m
../$projectnameforgin.Profiler_output.csv -o
${projectnameforgin}.${sampler}_${modelType}_${promptLetter}_${editCount}_output.csv -h ~/.sdkman/candidates/maven/current
-timeoutMS 10000 -et gin.edit.llm.LLMReplaceStatement -pn $editCount -pt simple-prompt.txt -mt $modelType -mo 300 -oaik demo

> modelType='mistral';
> java -Dtinylog.level=trace -cp ../gin-llm/build/gin.jar gin.util.$sampler -j -p $projectnameforgin -d . -m
../$projectnameforgin.Profiler_output.csv -o
${projectnameforgin}.${sampler}_${modelType}_${promptLetter}_${editCount}_output.csv -h ~/.sdkman/candidates/maven/current
-timeoutMS 10000 -et gin.edit.llm.LLMReplaceStatement -pn $editCount -pt simple-prompt.txt -mt $modelType -mo 300
```
```

Generating tests and Profiling

Generate new test cases

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

Profile a test suite

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

Results written to `profiler_output.csv`.

Example Profiler Output

	A	B	C	D	E
1	Project	Rank	Method	Count	Tests
2	jcodec	1	org.jcodec.codecs.h264.decode.deblock.DeblockingFilter.filterBs(int,int,int,byte[],byte[],int,int,int,int,int,int,boolean)	1077	org.jcodec.api.FrameGrabTest.testFrameGrab [],org.jcodec.api.Issue136Test.testFrameGrabDoesNotThrowException [],org.jcodec.api.Issue180Test.testFrameGrabDoesNotThrowException [],org.jcodec.api.Issue379H264LTRTest.testFrameGrabCanExtractOrientation [],org.jcodec.api.Issue383Test.testFrameGrabCanExtractOrientation [],org.jcodec.api.TranscoderTest.testCanFilterVideoAndCopyAudio [],org.jcodec.codecs.h264.H264EncoderTest.testEncodeDecode [],org.jcodec.codecs.h264.MBlock8x8Test.testMBlockCABACStrict1 [],org.jcodec.codecs.h264.MBlock8x8Test.testMBlockCABACStrict2 [],org.jcodec.codecs.h264.MBlock8x8Test.testMBlockCABACStrict3 [],org.jcodec.codecs.h264.MBlock8x8Test.testMBlockCABACStrict4 [],org.jcodec.codecs.h264.MBlock8x8Test.testMBlockCAVLCStrict2 [],org.jcodec.codecs.h264.MBlock8x8Test.testMBlockCAVLCStrict3 [],org.jcodec.codecs.h264.MBlock8x8Test.testMBlockCAVLCStrict4 [],org.jcodec.codecs.h264.MacroblockBiDecodingTest.testB16x16CABAC [],org.jcodec.codecs.h264.MacroblockBiDecodingTest.testB16x8CABAC [],org.jcodec.codecs.h264.MacroblockBiDecodingTest.testB4x8CABAC [],org.jcodec.codecs.h264.MacroblockBiDecodingTest.testB8x16CAVLC [],org.jcodec.codecs.h264.MacroblockBiDecodingTest.testB8x4CAVLC [],org.jcodec.codecs.h264.MacroblockBiDecodingTest.testB8x8CAVLC [],org.jcodec.codecs.h264.MacroblockBiDecodingTest.testSpatialDirect [],org.jcodec.codecs.h264.MacroblockI16x16DecodingTest.testMBlockCABAC3 [],org.jcodec.codecs.h264.MacroblockINxNDecodingTest.testMBlockCABAC4 [],org.jcodec.codecs.h264.MacroblockINxNDecodingTest.testMBlockCAVLC3 [],org.jcodec.codecs.h264.MacroblockIntraMixedDecodingTest.testMBlockCABAC3 [],org.jcodec.codecs.h264.MacroblockIntraMixedDecodingTest.testMBlockCAVLC4 [],org.jcodec.codecs.h264.MacroblockIntraMixedDecodingTest.testMBlockShit [],org.jcodec.codecs.h264.ScalingListTest.testScalingList [],org.jcodec.codecs.h264.conformance.ConformanceTest.testMp4Container [],org.jcodec.codecs.h264.conformance.ConformanceTest.testNoContainer []
3	jcodec	2	org.jcodec.codecs.h264.encode.MotionEstimator.estimateQPix(Picture,byte[],int[],int,int)	482	org.jcodec.api.SequenceEncoderTest.testRuns [],org.jcodec.api.TranscoderTest.testCanFilterVideoAndCopyAudio [],org.jcodec.codecs.h264.H264EncoderTest.testEncodeDecode [],org.jcodec.codecs.h264.encode.MotionEstimatorTest.testMotionEstimator []
4	jcodec	3	org.jcodec.codecs.h264.encode.MBEncoderHelper.takeSafe(byte[],int,int,int,int,byte[],int,int)	453	org.jcodec.api.SequenceEncoderTest.testRuns [],org.jcodec.api.TranscoderTest.testCanFilterVideoAndCopyAudio [],org.jcodec.codecs.h264.H264EncoderTest.testBufferOverflowImage [],org.jcodec.codecs.h264.H264EncoderTest.testEncodeDecode [],org.jcodec.codecs.h264.H264EncoderTest.testEncodeYuv420J [],org.jcodec.codecs.h264.encode.IntraPredEstimatorTest.testLumaPred4x4 [],org.jcodec.codecs.h264.encode.MotionEstimatorTest.testMotionEstimator []
5	jcodec	4	org.jcodec.codecs.h264.encode.IntraPredEstimator.getLumaPred4x4(Picture,EncodingContext,int,int,int)	444	org.jcodec.api.SequenceEncoderTest.testRuns [],org.jcodec.api.TranscoderTest.testCanFilterVideoAndCopyAudio [],org.jcodec.codecs.h264.H264EncoderTest.testBufferOverflowImage [],org.jcodec.codecs.h264.H264EncoderTest.testEncodeDecode [],org.jcodec.codecs.h264.H264EncoderTest.testEncodeYuv420J [],org.jcodec.codecs.h264.encode.IntraPredEstimatorTest.testLumaPred4x4 []
6	jcodec	5	org.jcodec.codecs.h264.decode.deblock.DeblockingFilter.filterBlockEdgeHoris(Picture,int,int,int,int,int,int,int)	443	org.jcodec.api.FrameGrabTest.testFrameGrab [],org.jcodec.api.Issue136Test.testFrameGrabDoesNotThrowException [],org.jcodec.api.Issue180Test.testFrameGrabDoesNotThrowException [],org.jcodec.api.Issue379H264LTRTest.testFrameGrabCanExtractOrientation [],org.jcodec.api.Issue383Test.testFrameGrabCanExtractOrientation [],org.jcodec.api.TranscoderTest.testCanFilterVideoAndCopyAudio [],org.jcodec.codecs.h264.H264EncoderTest.testEncodeDecode [],org.jcodec.codecs.h264.MBlock8x8Test.testMBlockCABACStrict3 [],org.jcodec.codecs.h264.MacroblockBiDecodingTest.testB16x8CABAC [],org.jcodec.codecs.h264.MacroblockBiDecodingTest.testB8x8CABAC [],org.jcodec.codecs.h264.MacroblockBiDecodingTest.testBiDirectSpatialNoInference [],org.jcodec.codecs.h264.MacroblockBiDecodingTest.testLongTermCABAC [],org.jcodec.codecs.h264.MacroblockBiDecodingTest.testMixedBiRefCAVLC [],org.jcodec.codecs.h264.MacroblockINxNDecodingTest.testMBlockCAVLC3 [],org.jcodec.codecs.h264.MacroblockIntraMixedDecodingTest.testMBlockCABAC3 [],org.jcodec.codecs.h264.MacroblockIntraMixedDecodingTest.testMBlockCAVLC1 [],org.jcodec.codecs.h264.MacroblockIntraMixedDecodingTest.testMBlockShit [],org.jcodec.codecs.h264.ScalingListTest.testScalingList [],org.jcodec.codecs.h264.conformance.ConformanceTest.testMp4Container [],org.jcodec.codecs.h264.conformance.ConformanceTest.testNoContainer []
7					org.jcodec.api.FrameGrabTest.testFrameGrab [],org.jcodec.api.Issue136Test.testFrameGrabDoesNotThrowException [],org.jcodec.api.Issue180Test.testFrameGrabDoesNotThrowException [],org.jcodec.api.Issue379H264LTRTest.testFrameGrabCanExtractOrientation [],org.jcodec.api.Issue383Test.testFrameGrabCanExtractOrientation [],org.jcodec.api.TranscoderTest.testCanFilterVideoAndCopyAudio [],org.jcodec.codecs.h264.H264EncoderTest.testEncodeDecode [],org.jcodec.codecs.h264.MBlock8x8Test.testMBlockCAVLCStrict4 [],org.jcodec.codecs.h264.MacroblockBiDecodingTest.testB8x16CABAC [],org.jcodec.codecs.h264.MacroblockBiDecodingTest.testBiDirectSpatialNoInferenceCAVLC

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
```

- 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
```

Gin papers

Available from https://github.com/gintool/gin

Cite these:

GECCO 2019

Gin: Genetic Improvement Research Made Easy

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

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

Brad Alexander
School of Computer Science
University of Adelaide
Adelaide, Australia
brad@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

GECCO 2017

GI in No Time

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

Overview

- **Introduction: why GI? And basic principles**
- **Challenges and open research questions**
- **Case study: fixing bugs**
- **The human perspective**
- **Noteworthy papers, and connections to other topics**
- **Demonstration: Gin**
- **Summary and Q&A**

GI Workshop

The 13th International Workshop on Genetic Improvement @ICSE 2024

- Held on 16 April
- Keynote from Shin Yoo and Tutorial from Aymeric Blot
- 7 accepted papers
- Future workshops http://geneticimprovementofsoftware.com

Summary and Q&A

Questions?

Sæmundur (Saemi) Haraldsson <soh@cs.stir.ac.uk>

John Woodward <j.woodward@qmul.ac.uk>

Alexander (Sandy) Brownlee <alexander.brownlee@stir.ac.uk>

Markus Wagner <markus.wagner@monash.edu>

Latest version of slides at https://cs.stir.ac.uk/~sbr/files/GI_tutorial_GECCO_2024.pdf

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 Vilmundur 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

Petke, J., Haraldsson, S. O., Harman, M., Langdon, W. B., White, D. R., & Woodward, J. R. (2017). Genetic improvement of software: a comprehensive survey. IEEE Transactions on Evolutionary Computation, 22(3), 415-432. DOI: 10.1109/TEVC.2017.2693219

J. Petke, 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. Petke, 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/10.1145/3196398.3196442

E. R. Winter et al., "Let's Talk With Developers, Not About Developers: A Review of Automatic Program Repair Research," in IEEE Transactions on Software Engineering, doi: https://10.1109/TSE.2022.3152089

V. Nowack et al., "Expanding Fix Patterns to Enable Automatic Program Repair," 2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE), 2021, pp. 12-23, doi: https://10.1109/ISSRE52982.2021.00015 .

E. Winter et al., "How do Developers Really Feel About Bug Fixing? Directions for Automatic Program Repair," in IEEE Transactions on Software Engineering, vol. 49, no. 4, pp. 1823-1841, 1 April 2023, doi: 10.1109/TSE.2022.3194188.

G. B. Saemundsdottir, and S. O. Haraldsson, 2024. “Large Language Models as All-in-One Operators for Genetic Improvement.”, to appear in Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '24). ACM, New York, NY, USA, 1501-1508. URL: https://doi.org/10.1145/3638530.3654408

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 Vilmundur 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

Petke, J., Haraldsson, S. O., Harman, M., Langdon, W. B., White, D. R., & Woodward, J. R. (2017). Genetic improvement of software: a comprehensive survey. IEEE Transactions on Evolutionary Computation, 22(3), 415-432. DOI: 10.1109/TEVC.2017.2693219

J. Petke, 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. Petke, 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/10.1145/3196398.3196442

E. R. Winter et al., "Let's Talk With Developers, Not About Developers: A Review of Automatic Program Repair Research," in IEEE Transactions on Software Engineering, doi: https://10.1109/TSE.2022.3152089

V. Nowack et al., "Expanding Fix Patterns to Enable Automatic Program Repair," 2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE), 2021, pp. 12-23, doi: https://10.1109/ISSRE52982.2021.00015 .

E. Winter et al., "How do Developers Really Feel About Bug Fixing? Directions for Automatic Program Repair," in IEEE Transactions on Software Engineering, vol. 49, no. 4, pp. 1823-1841, 1 April 2023, doi: 10.1109/TSE.2022.3194188.

G. B. Saemundsdottir, and S. O. Haraldsson, 2024. “Large Language Models as All-in-One Operators for Genetic Improvement.”, to appear in Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '24). ACM, New York, NY, USA, 1501-1508. URL: https://doi.org/10.1145/3638530.3654408