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

Alexander E.I. Brownlee, Sæmundur Ó. Haraldsson, John R. Woodward

UNIVERSITY of STIRLING

Operational Research
Queen Mary University of London

PPSN 2020

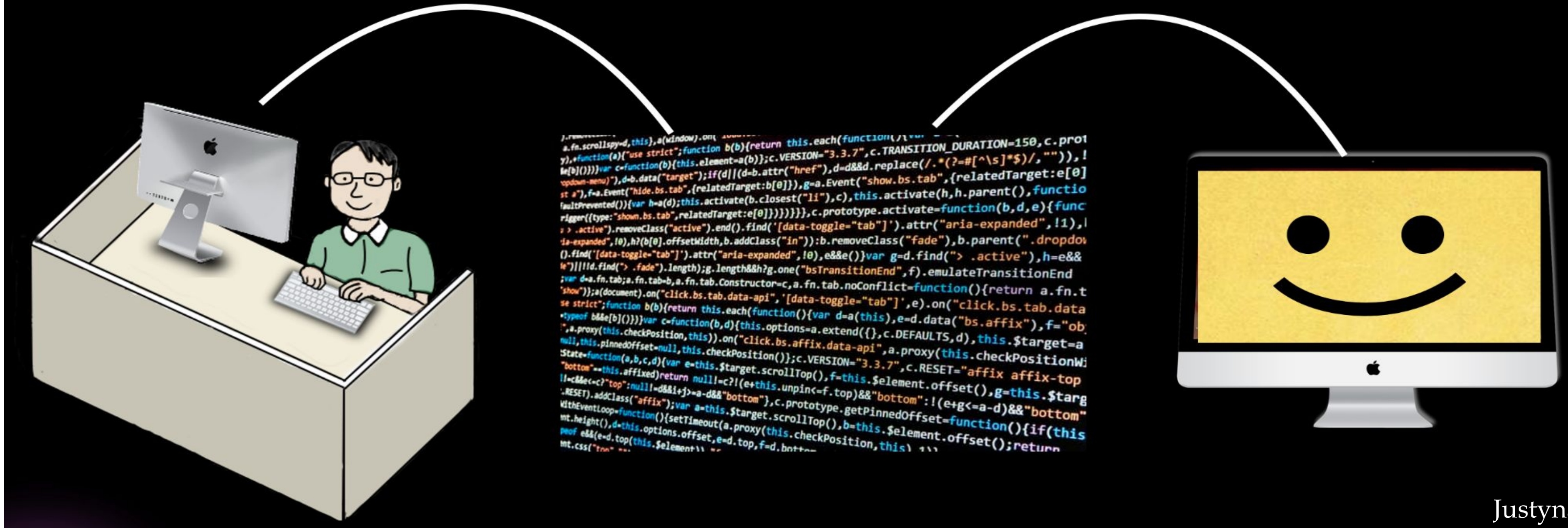
Overview

- **Introduction**
- **Example: Fixing Bugs**
- **Getting involved**
- **Summary and Q&A**

Genetic Improvement of Software

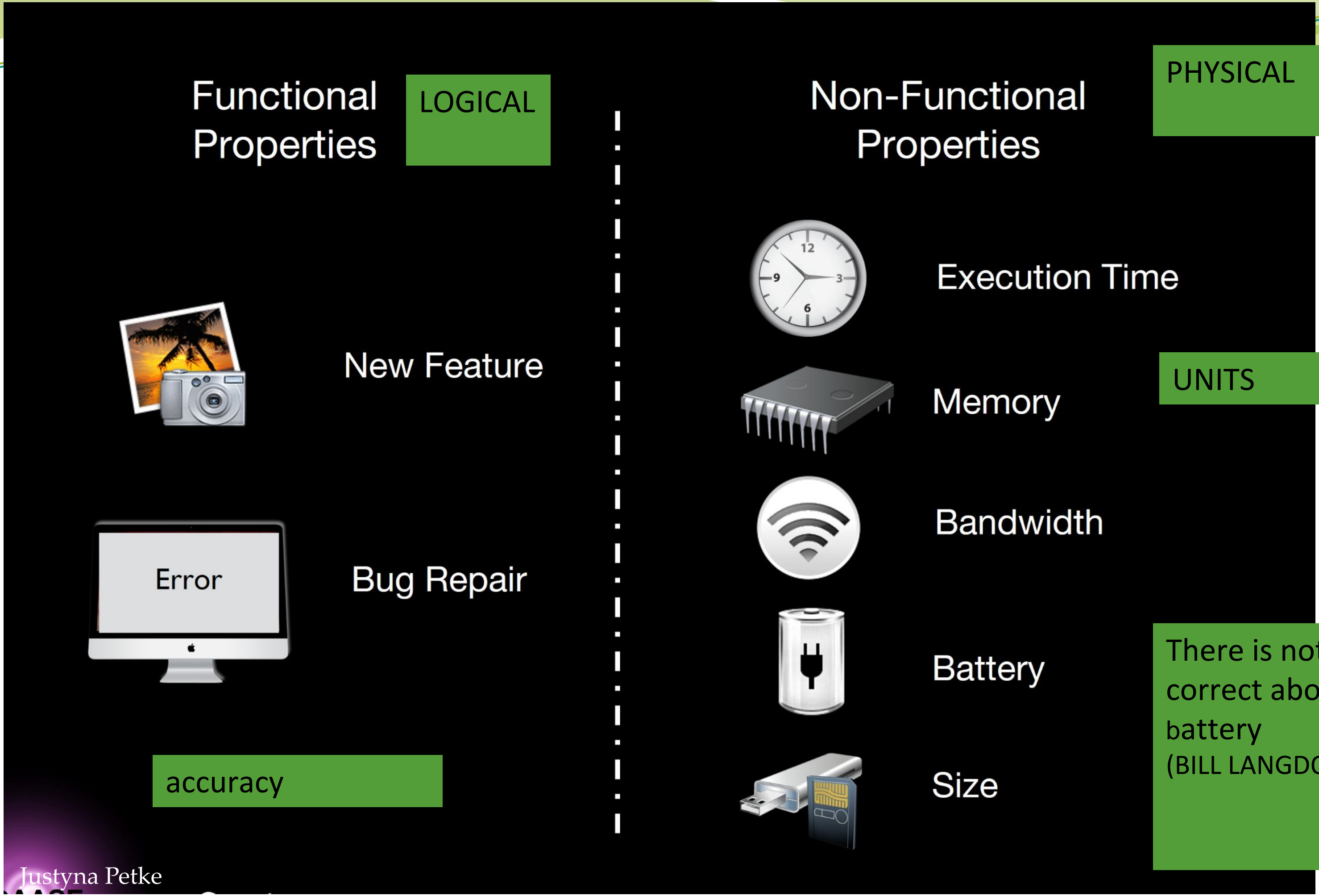
human writes code

computer improves it



```
a.fn.scrollspy=d,this},a(window).on(
y),+function(a){"use strict";function b(b){return this.each(function(){var
e[b]()}))}var c=function(b){this.element=a(b)};c.VERSION="3.3.7",c.TRANSITION_DURATION=150,c.prot
opdown-menu)"),d=b.data("target");if(d||(d=b.attr("href"),d=d&&d.replace(/.*(?=#[^\s]*$)/,"")),!
st a"),f=a.Event("hide.bs.tab",{relatedTarget:b[0]}),g=a.Event("show.bs.tab",{relatedTarget:e[0]
faultPrevented()){var h=a(d);this.activate(b.closest("li"),c),this.activate(h,h.parent(),functio
rigger({type:"shown.bs.tab",relatedTarget:e[0]})})}}},c.prototype.activate=function(b,d,e){func
> .active").removeClass("active").end().find('[data-toggle="tab"]').attr("aria-expanded",!1),
ia-expanded",!0),h?(b[0].offsetWidth,b.addClass("in")):b.removeClass("fade"),b.parent(".dropdow
().find('[data-toggle="tab"]').attr("aria-expanded",!0),e&&e()}var g=d.find("> .active"),h=e&&
e")||!!d.find("> .fade").length);g.length&&h?g.one("bsTransitionEnd",f).emulateTransitionEnd
;var d=a.fn.tab;a.fn.tab=b,a.fn.tab.Constructor=c,a.fn.tab.noConflict=function(){return a.fn.t
"show")};a(document).on("click.bs.tab.data-api",'[data-toggle="tab"]',e).on("click.bs.tab.data
se strict";function b(b){return this.each(function(){var d=a(this),e=d.data("bs.affix"),f="ob
typeof b&&e[b]()}))}var c=function(b,d){this.options=a.extend({},c.DEFAULTS,d),this.$target=a
",a.proxy(this.checkPosition,this)).on("click.bs.affix.data-api",a.proxy(this.checkPositionWi
null,this.pinnedOffset=null,this.checkPosition()};c.VERSION="3.3.7",c.RESET="affix affix-top
State=function(a,b,c,d){var e=this.$target.scrollTop(),f=this.$element.offset(),g=this.$targ
"bottom"==this.affixed)return null!=c?!(e+this.unpin<=f.top)&&"bottom":!(e+g<=a-d)&&"bottom"
!=c&&e<=c?"top":null!=d&&i+j>=a-d&&"bottom"},c.prototype.getPinnedOffset=function(){if(this
.RESET).addClass("affix");var a=this.$target.scrollTop(),b=this.$element.offset();return
WithEventLoop=function(){setTimeout(a.proxy(this.checkPosition,this),1)}
nt.height(),d=this.options.offset,e=d.top,f=d.bottom
eof e&&(e=d.top(this.$element))
nt.css("top",
```

Justyna Petke



Functional Properties

LOGICAL

Non-Functional Properties

PHYSICAL

- New Feature
- Bug Repair

accuracy

- Execution Time
- Memory
- Bandwidth
- Battery
- Size

UNITS

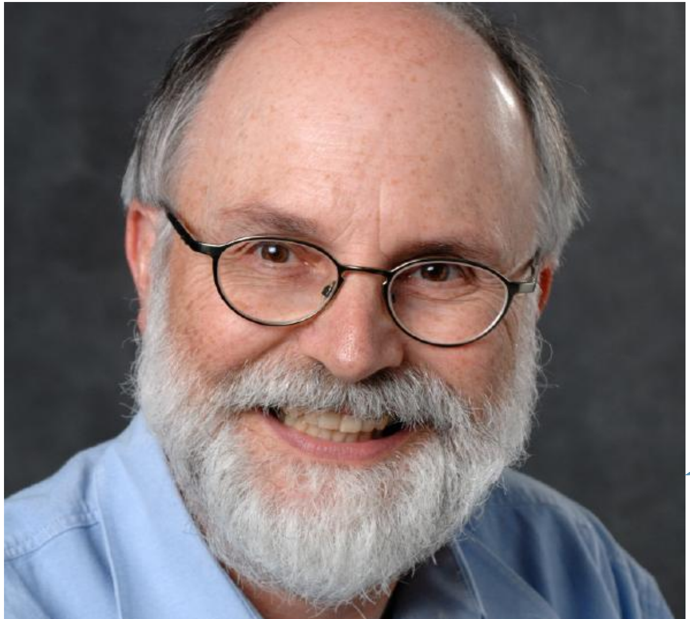
There is nothing correct about a flat battery
(BILL LANGDON)

Justyna Petke

What is Genetic Improvement

A wordy definition:

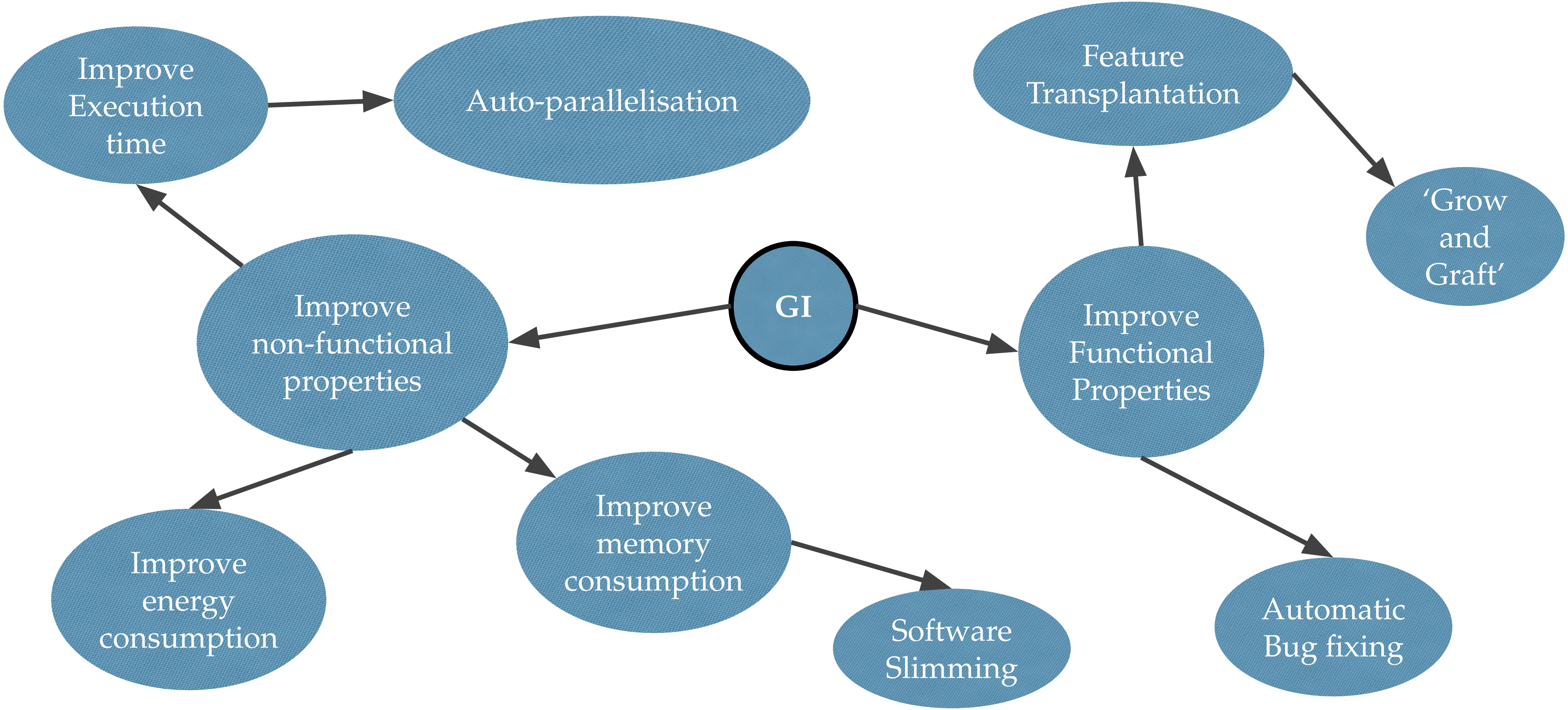
Genetic Improvement is the application of search-based (typically evolutionary) techniques to modify software with respect to some user-defined fitness measure.



It's just GP - BUT starting with a **nearly complete** program

[Wolfgang Banzhaf]

What is Genetic Improvement



GI

Improve non-functional properties

Improve Execution time

Auto-parallelisation

Improve energy consumption

Improve memory consumption

Software Slimming

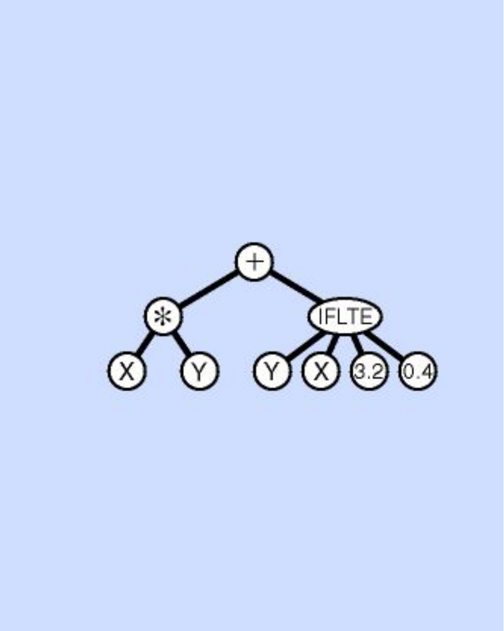
Improve Functional Properties

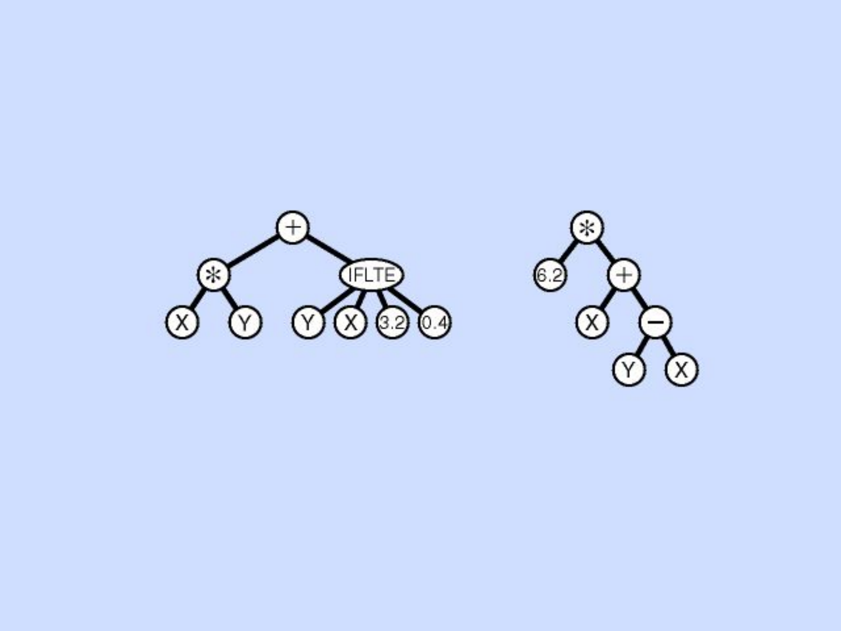
Feature Transplantation

'Grow and Graft'

Automatic Bug fixing

Genetic Programming overview





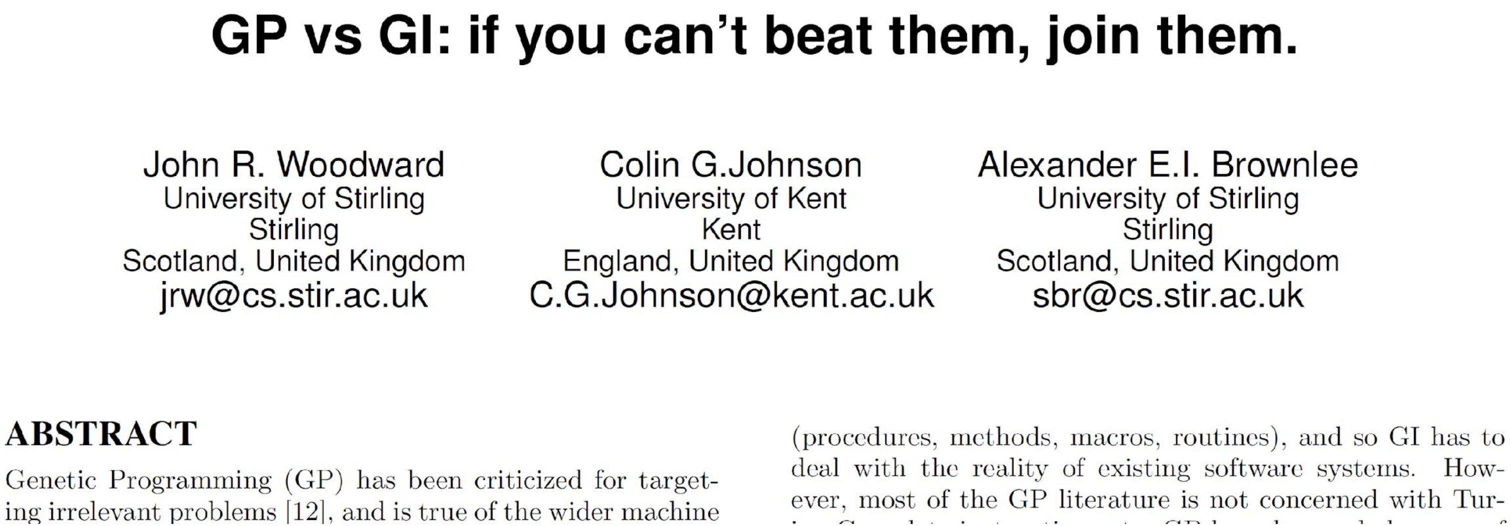
mutation

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

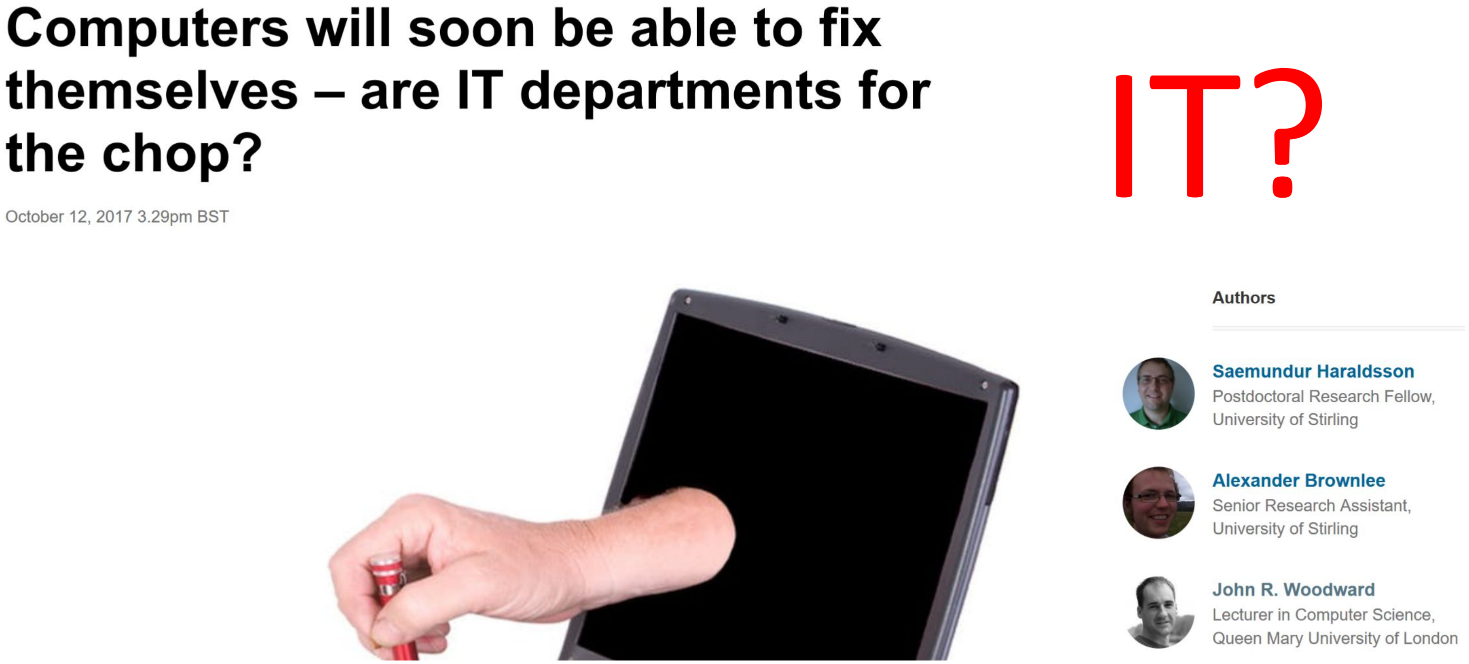
Genetic Programming (GP) has been criticized for targeting irrelevant problems [12], and is true of the wider machine

(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?

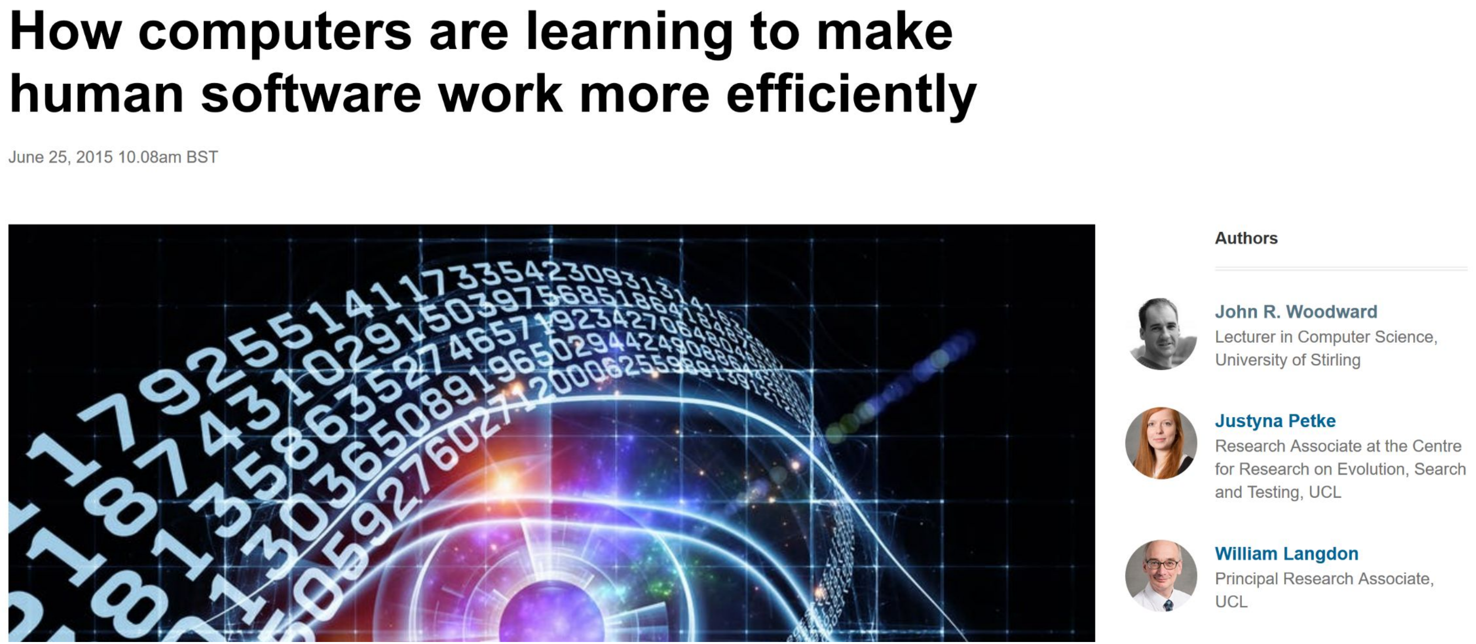
Authors

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

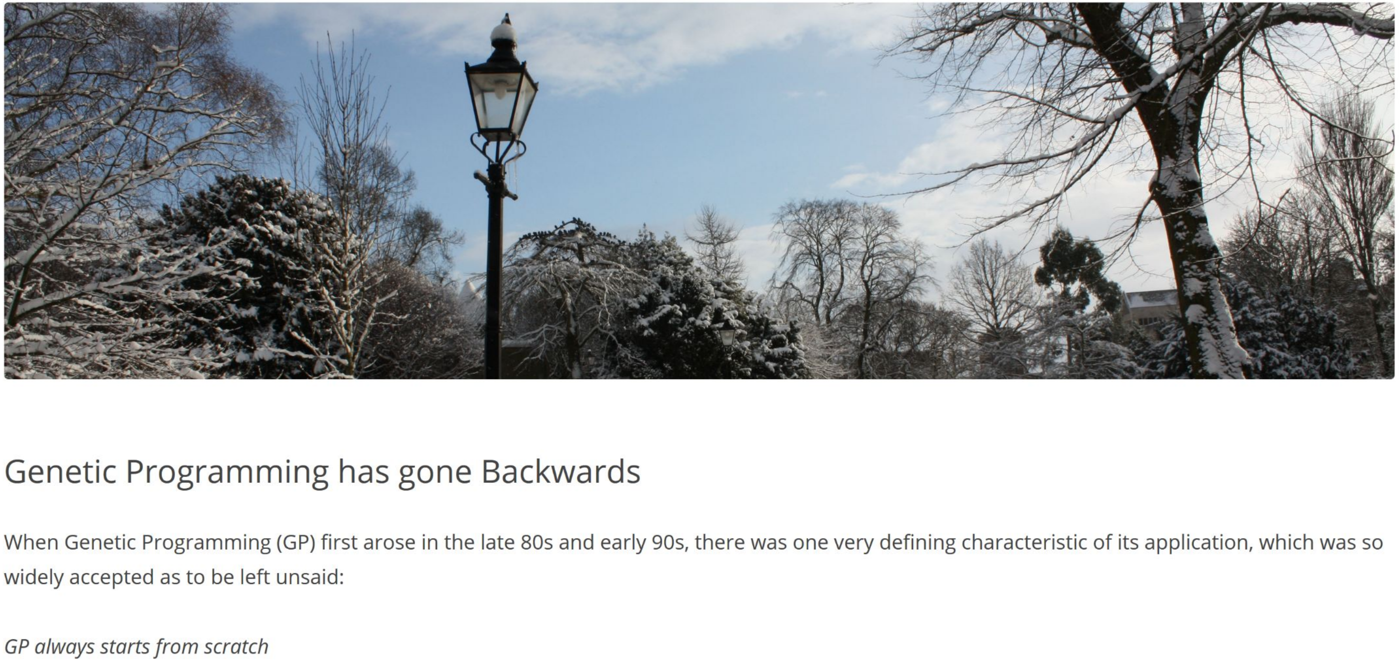
Authors

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/

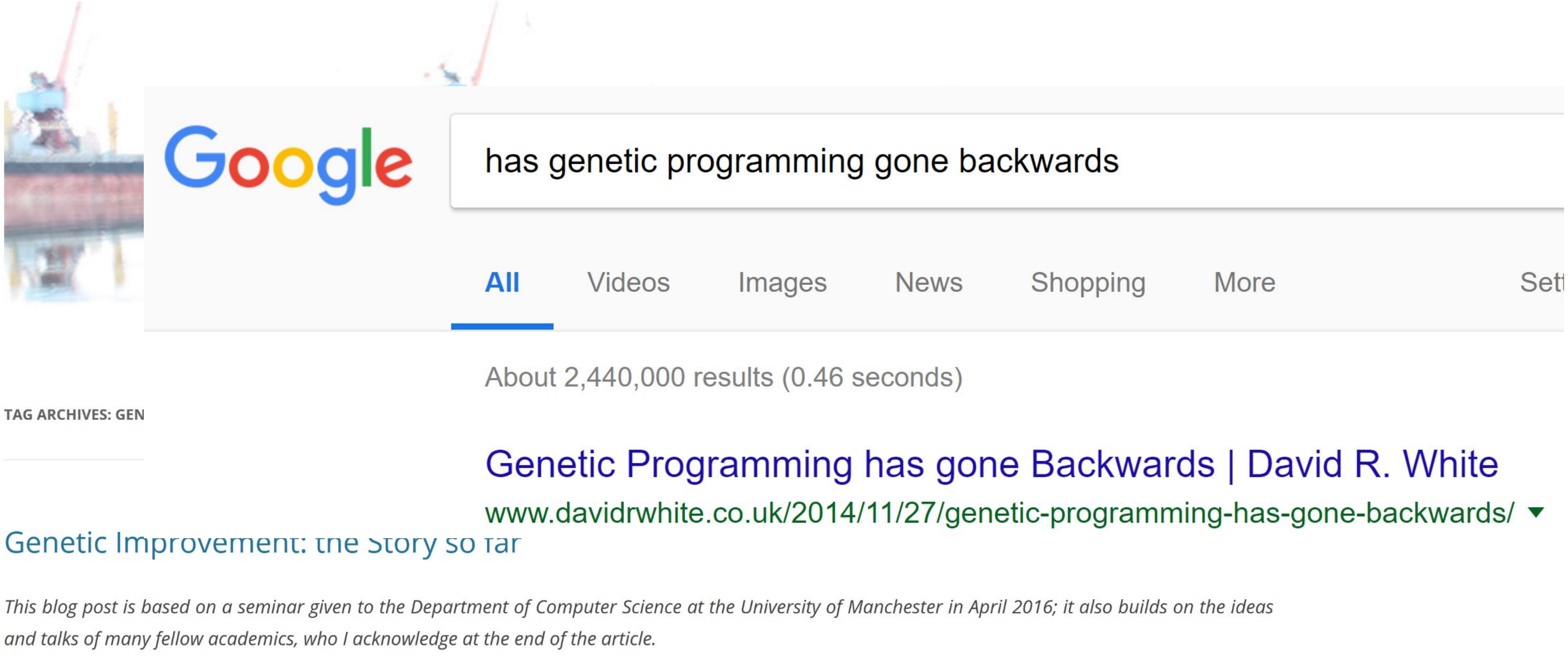


Genetic Programming has gone Backwards

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/



has genetic programming gone backwards

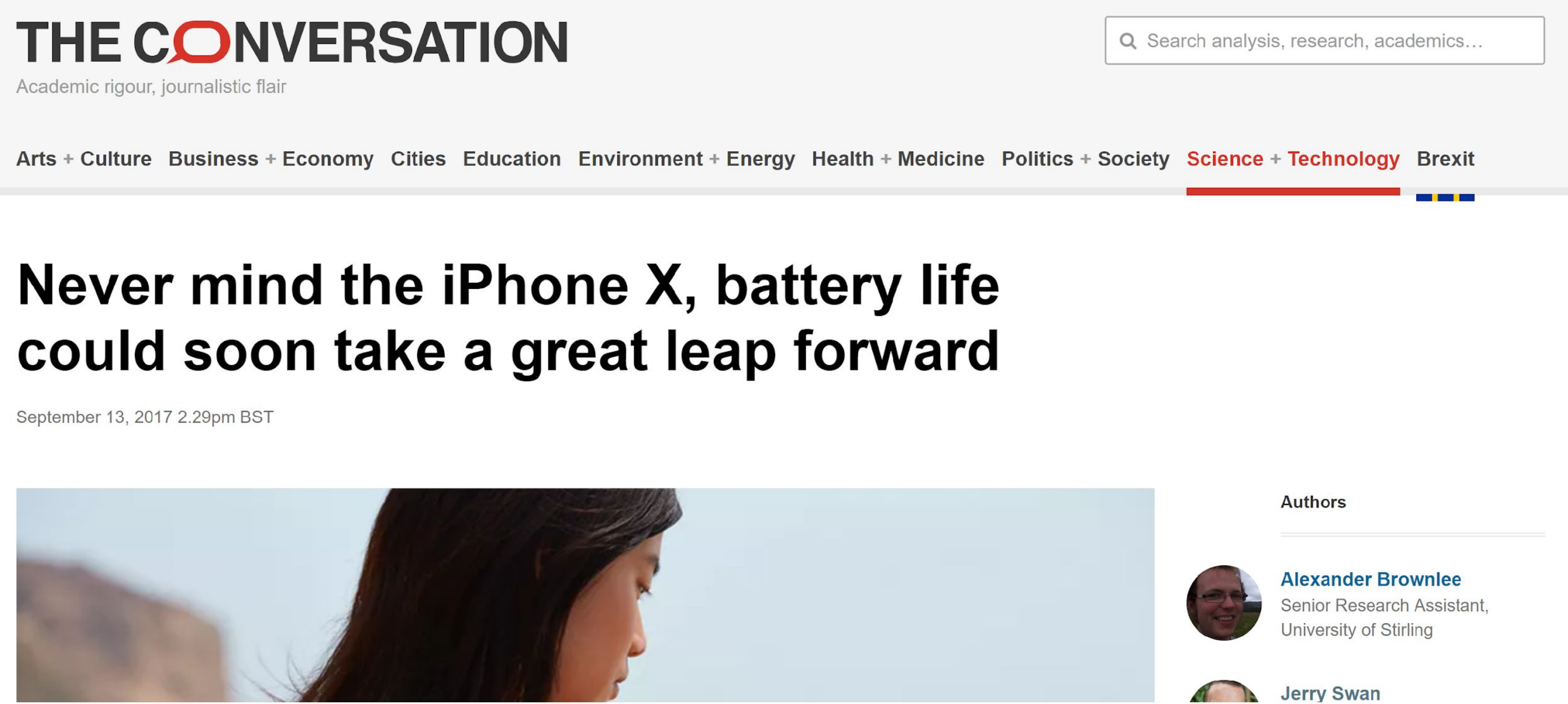
About 2,440,000 results (0.46 seconds)

Genetic Programming has gone Backwards | David R. White
www.davidrwhite.co.uk/2014/11/27/genetic-programming-has-gone-backwards/ ▾

TAG ARCHIVES: GEN

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

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]

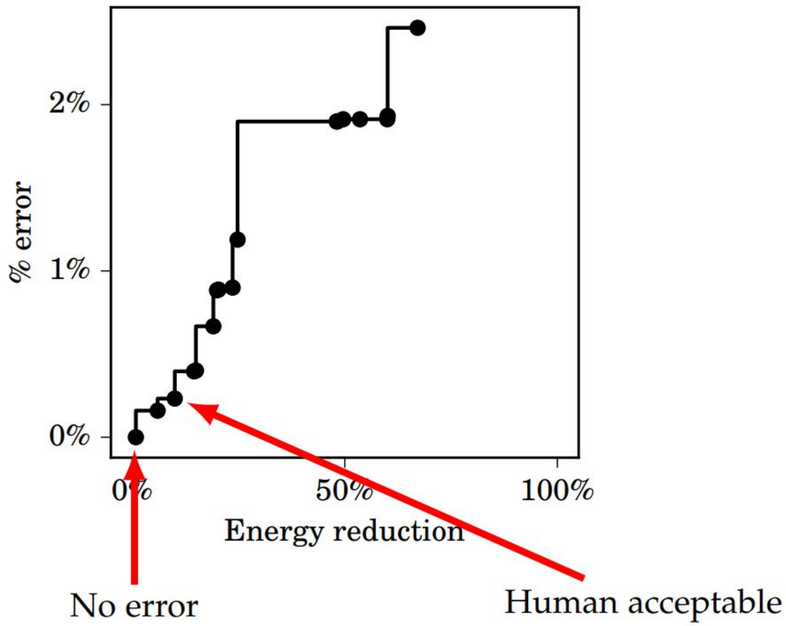
4. Errors, compile, halting (Langdon - discard)

Objectives

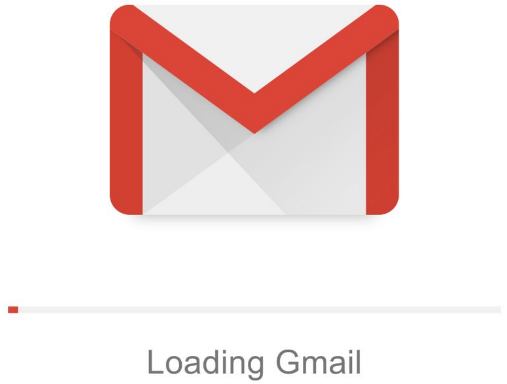
- Functional (**logical properties**)
  - Accuracy e.g. as in machine learning - FLOAT
  - Number of bugs – as measured against a set of test cases. BOOLEAN
  - New functionality – e.g.
- Non-functional (***physical*** **properties**)
  - Execution time
  - Energy (power consumption – peak/average)
  - 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 ir the solution space of programs for a specific programming problem.

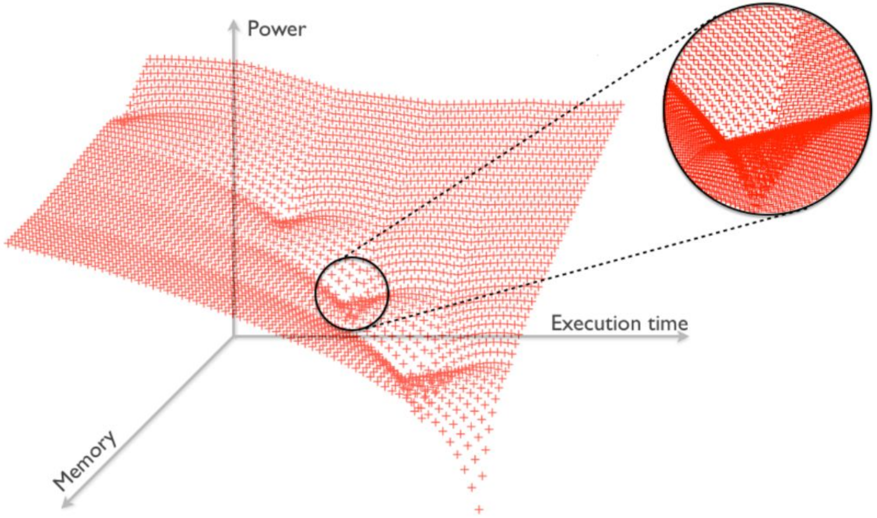


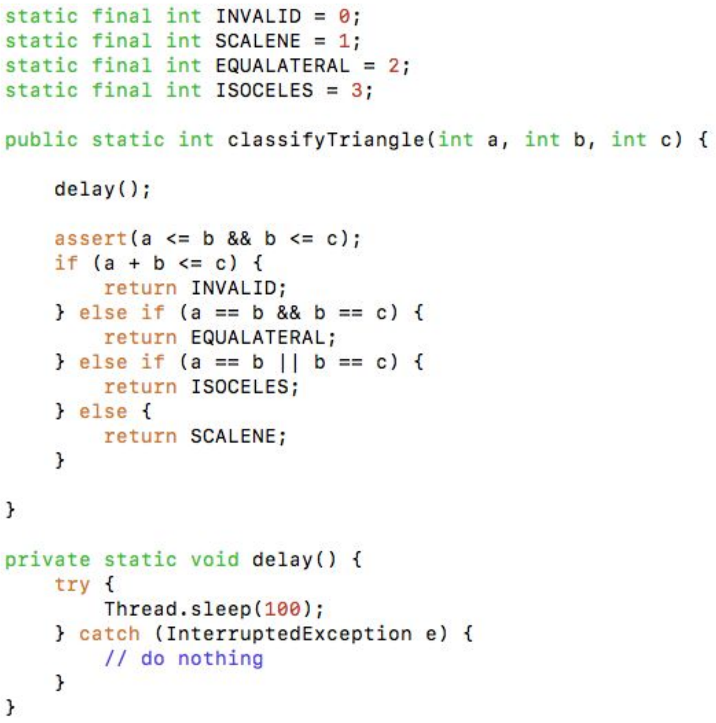
Figure 1: The GISMOE Pareto Program Surface

EDITS Operators – changes to programs

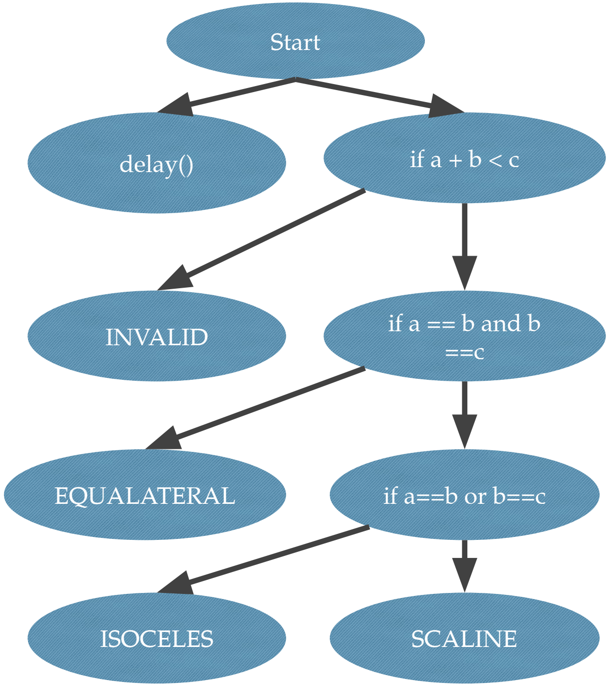
- Line level
- Single Character level
- Function/module level.
- AST – GIN, Gen-0-fix, genprog,
- Java – machine code – java byte code.

- LIST OF EDITS IS A PATCH.

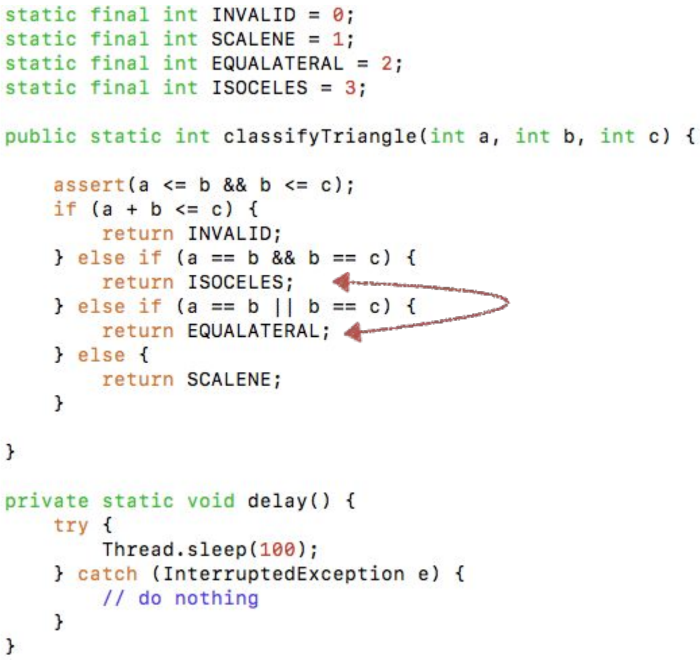
GI: An example of execution time optimisation



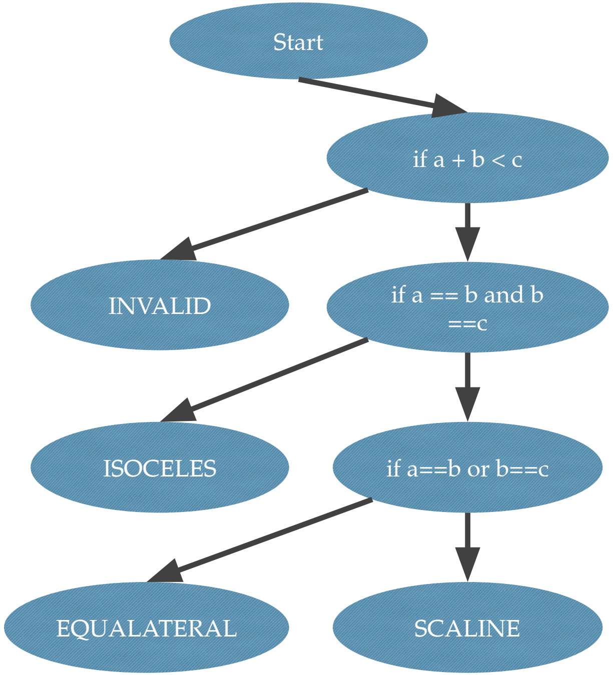
```
static final int INVALID = 0;
static final int SCALENE = 1;
static final int EQUALATERAL = 2;
static final int ISOCELES = 3;

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

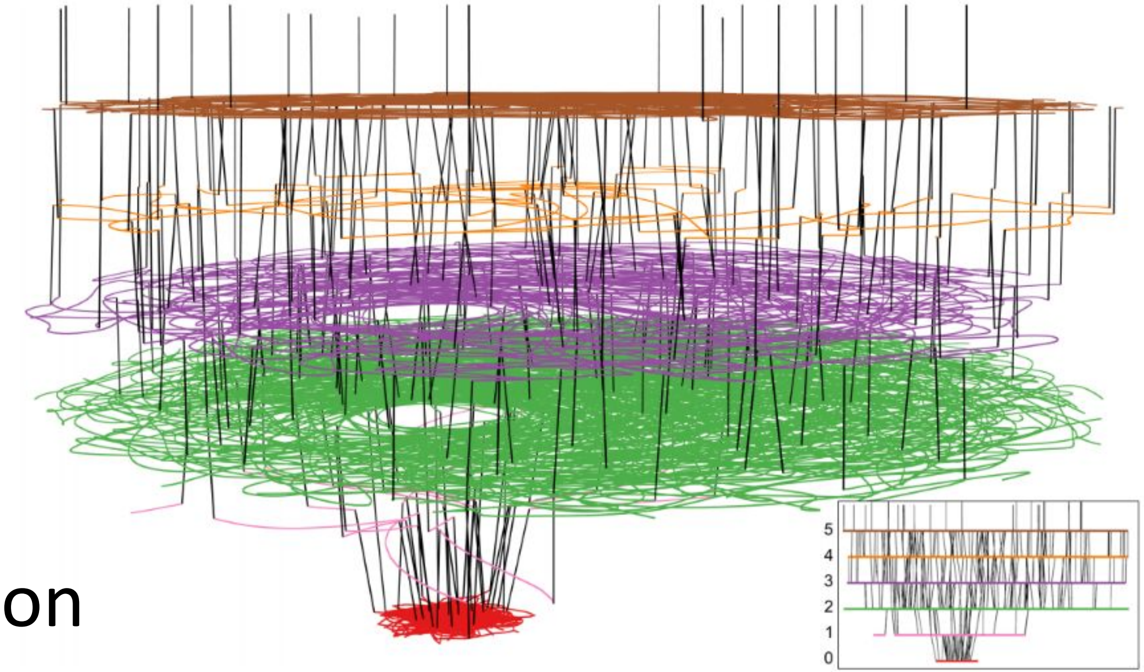
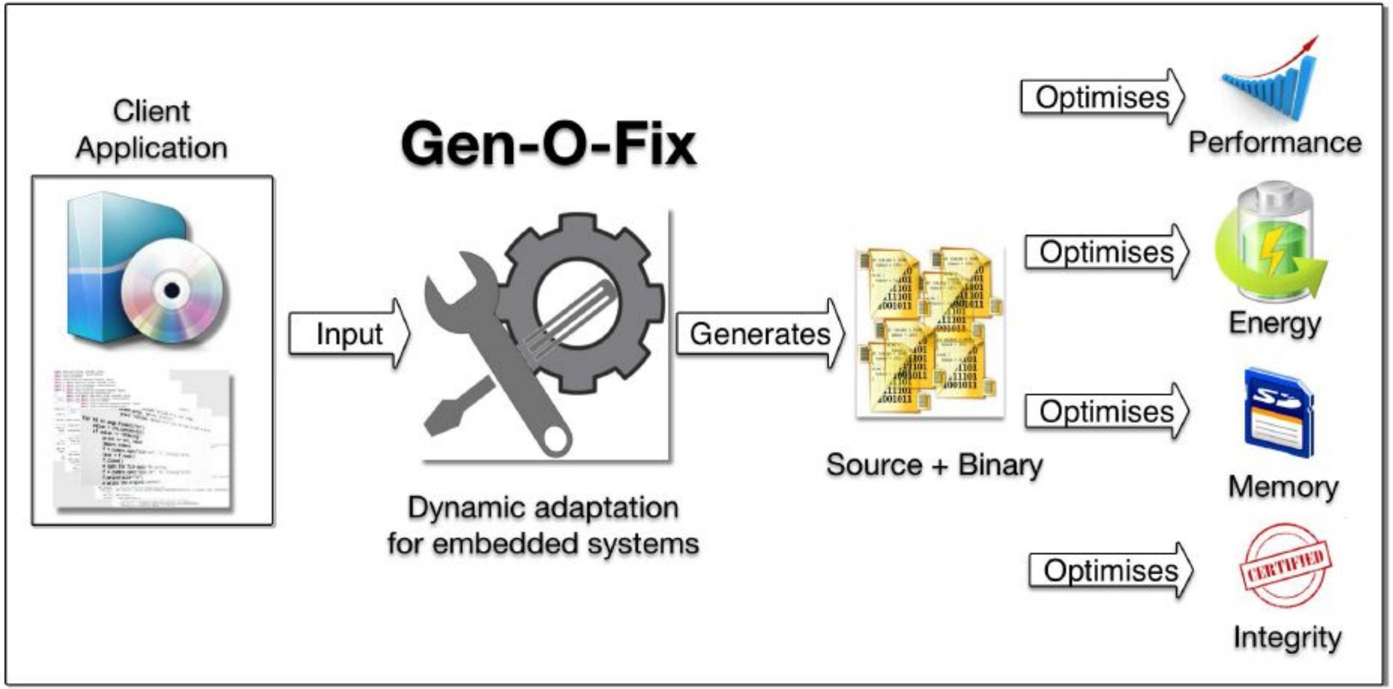


Fig. 1. Local optima network of the Triangle Program using 100 random starts (see Section 4.4). Edges are coloured if they start and end at the same fitness. Insert shows fitness levels edge on. Best (bottom) red 0 (pass all tests), pink 1 (fail only one test), green 2, purple 3, orange 4, brown 5.

structure

Hill climber

System Diagram for Gen-O-Fix



Gen-O-Fix: Abstract Syntax Trees

Main features of framework are

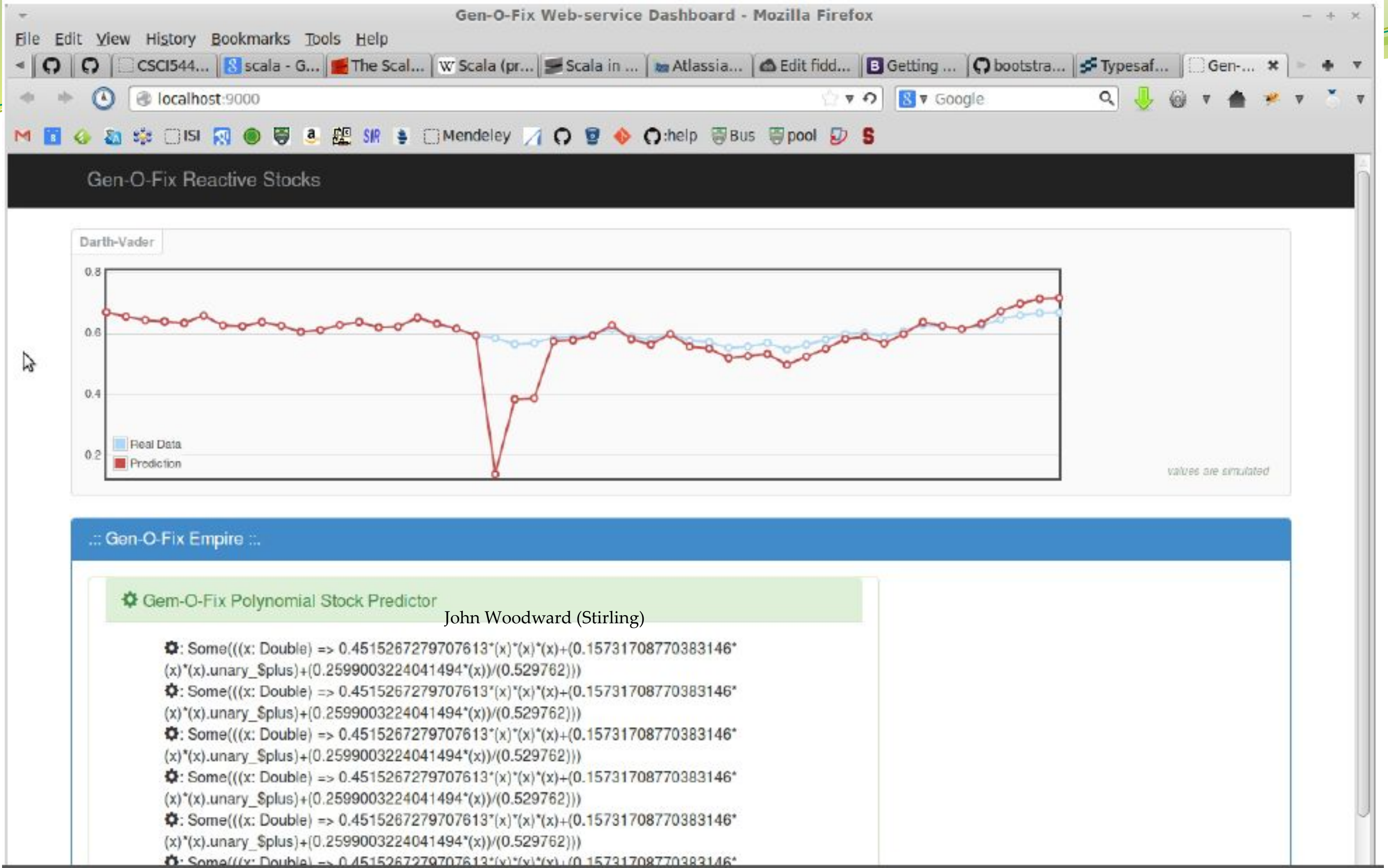
1. **Embedded** adaptively.
2. Minimal end-user requirements.
    1. Initial source code: **location** of Scala source code file containing a function
    2. Fitness function: providing a means of **evaluating the quality** of system
3. **Source to source transformations**
4. Operates on **ASTs** (i.e. 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
CSCI544... scala - G... The Scal... Scala (pr... Scala in ... Atlassia... Edit fidd... Getting ... bootstra... Typesaf... Gen-...
localhost:9000
Google
ISI Mendeley :help Bus pool
Darth-Vader
0.8
0.6
0.4
0.2
Real Data
Prediction
values are simulated

Gen-O-Fix Reactive Stocks

.:: Gen-O-Fix Empire ::.

Gem-O-Fix Polynomial Stock Predictor

```
: Some(((x: Double) => 0.4515267279707613*(x)*(x)*(x)+(0.1573170877038314 6*
(x)*(x).unary_$plus)+(0.259900322404 1494*(x))/(0.529762)))
: Some(((x: Double) => 0.4515267279707613*(x)*(x)*(x)+(0.1573170877038314 6*
(x)*(x).unary_$plus)+(0.259900322404 1494*(x))/(0.529762)))
: Some(((x: Double) => 0.4515267279707613*(x)*(x)*(x)+(0.1573170877038314 6*
(x)*(x).unary_$plus)+(0.259900322404 1494*(x))/(0.529762)))
: Some(((x: Double) => 0.4515267279707613*(x)*(x)*(x)+(0.1573170877038314 6*
(x)*(x).unary_$plus)+(0.259900322404 1494*(x))/(0.529762)))
: Some(((x: Double) => 0.4515267279707613*(x)*(x)*(x)+(0.1573170877038314 6*
(x)*(x).unary_$plus)+(0.259900322404 1494*(x))/(0.529762)))
: Some(((x: Double) => 0.4515267279707613*(x)*(x)*(x)+(0.1573170877038314 6*
```

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

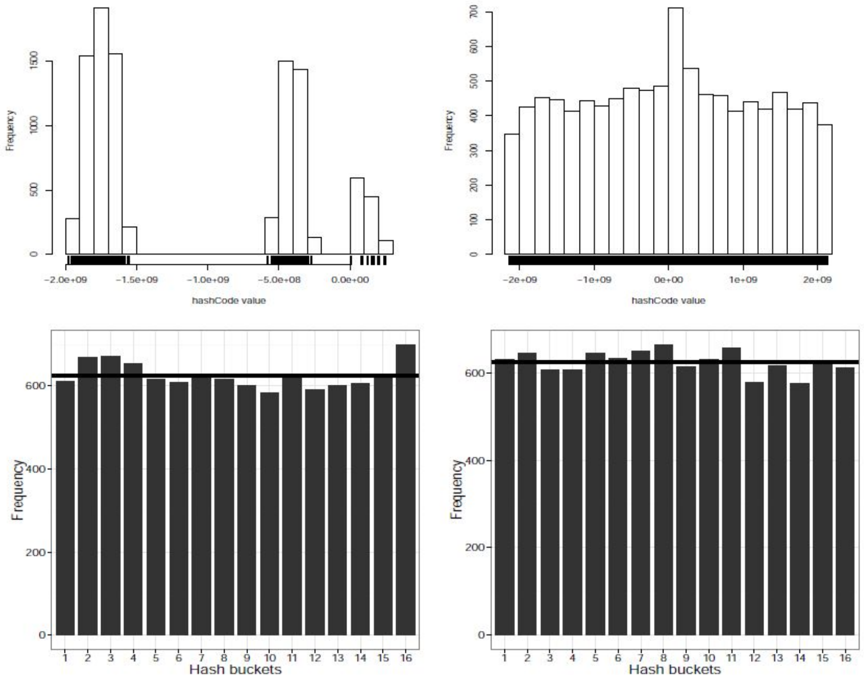


Fig. 1: The distribution of the hashcode values (top) and the distribution of the created objects in hash buckets (bottom), generated by the Apache commons (left) and the evolved function (right)

Overview

- Introduction
- Example: Fixing Bugs
- Getting involved
- Summary and Q&A

Fixing Bugs and other examples

Saemundur O. Haraldsson

- Fixing bugs
- Making software faster

FIXIE

Ref.:
EP/S005730/1

EPSRC
Engineering and Physical Sciences
Research Council

Lancaster University

Operational Research
Queen Mary University of London

Brunel
University
London

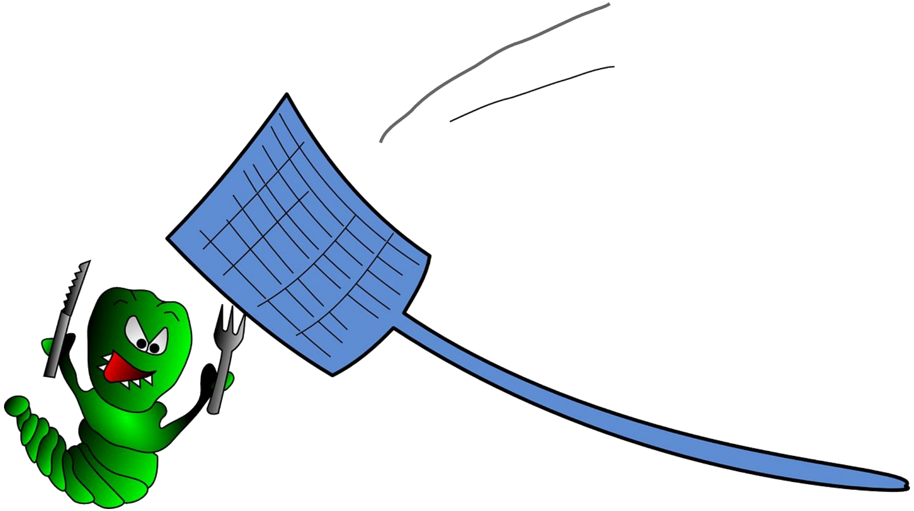
Fixing bugs

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

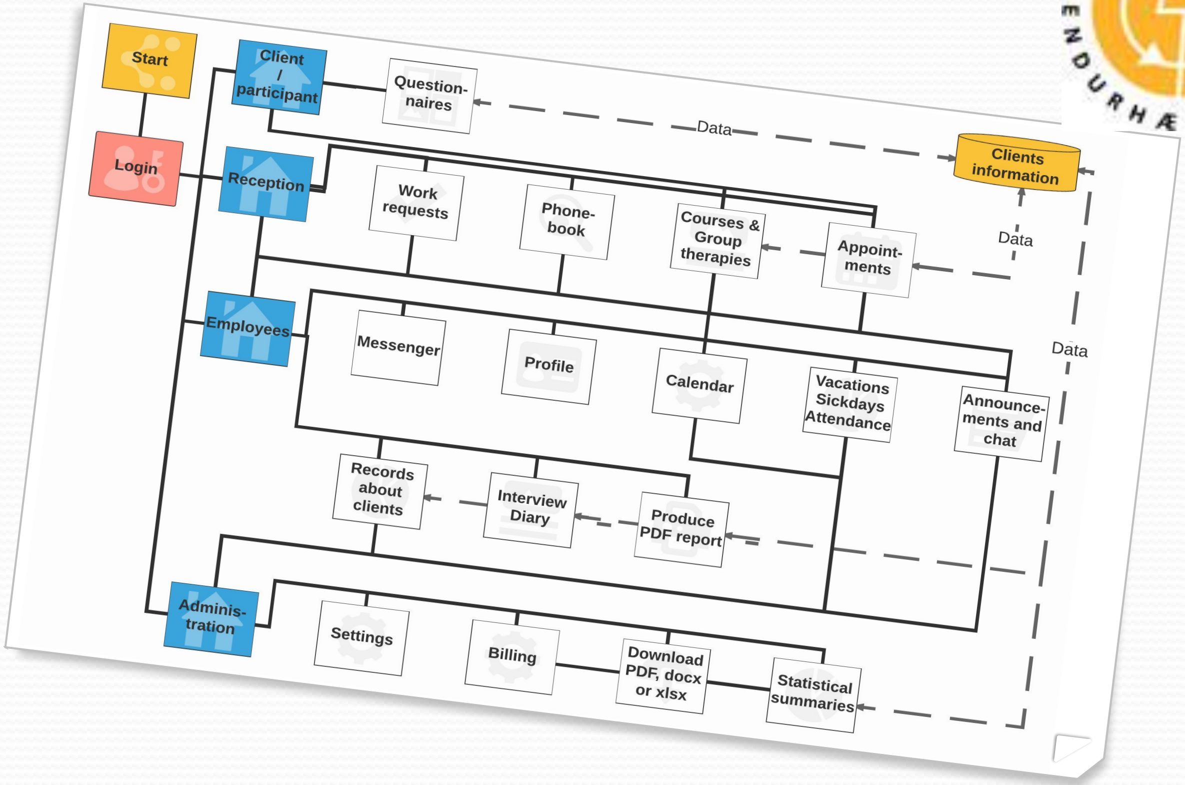
S. O. Haraldsson, J. R. Woodward and A. I. E. Brownlee, "The Use of Automatic Test Data Generation for Genetic Improvement in a Live System," 2017 IEEE/ACM 10th International Workshop on Search-Based Software Testing (SBST), Buenos Aires, 2017, pp. 28-31. DOI: https://10.1109/SBST.2017.10

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



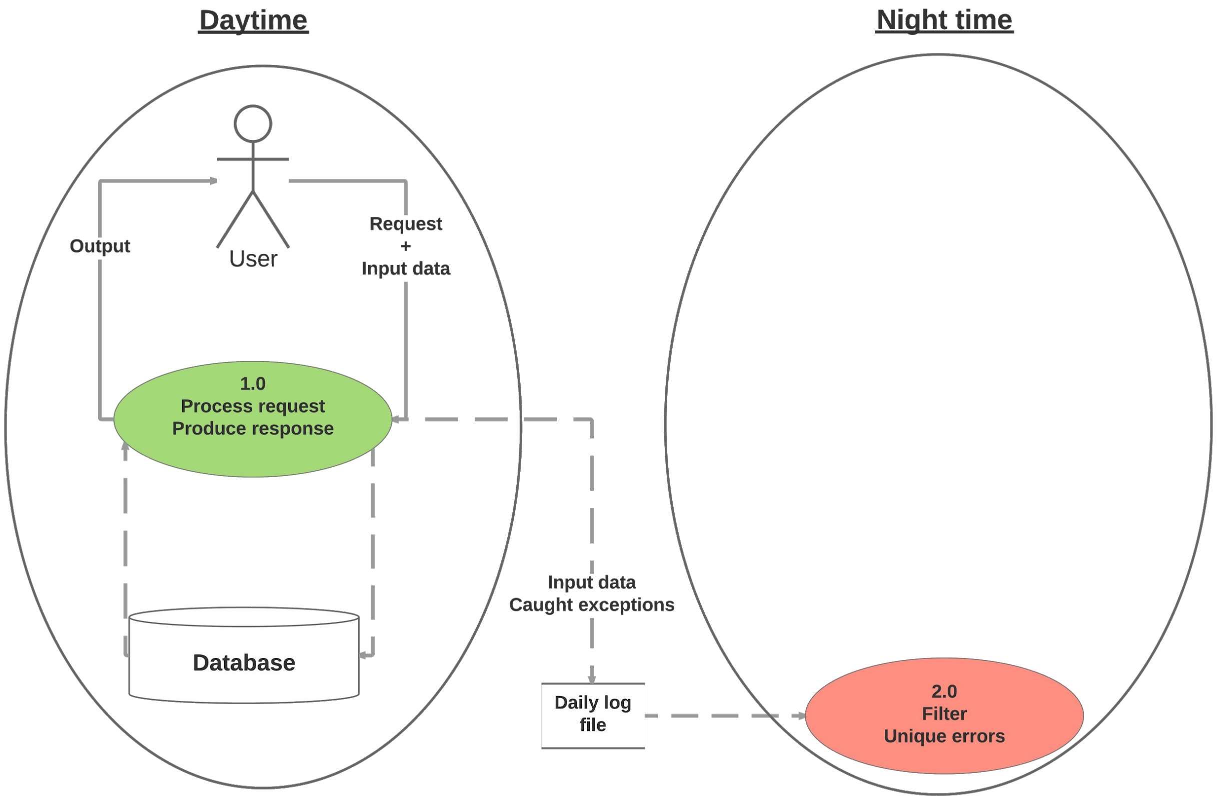
Janus Manager

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



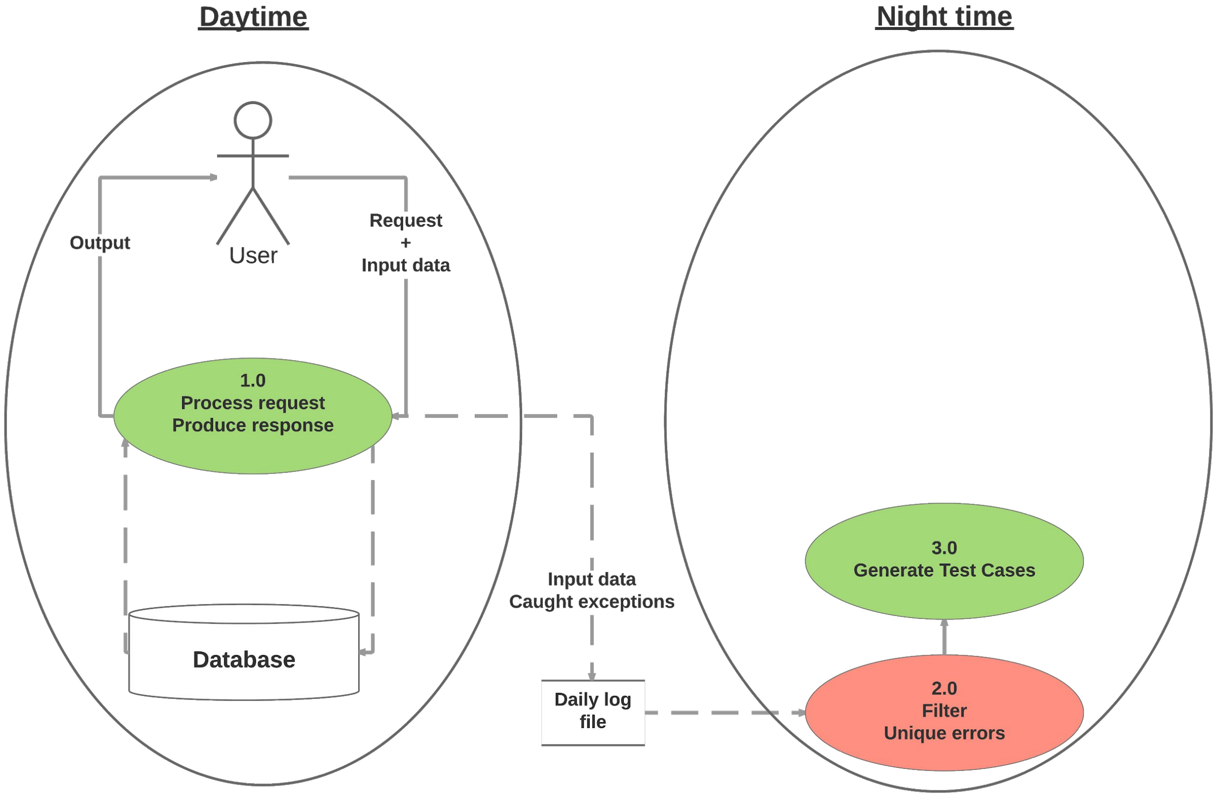
When last user logs out

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



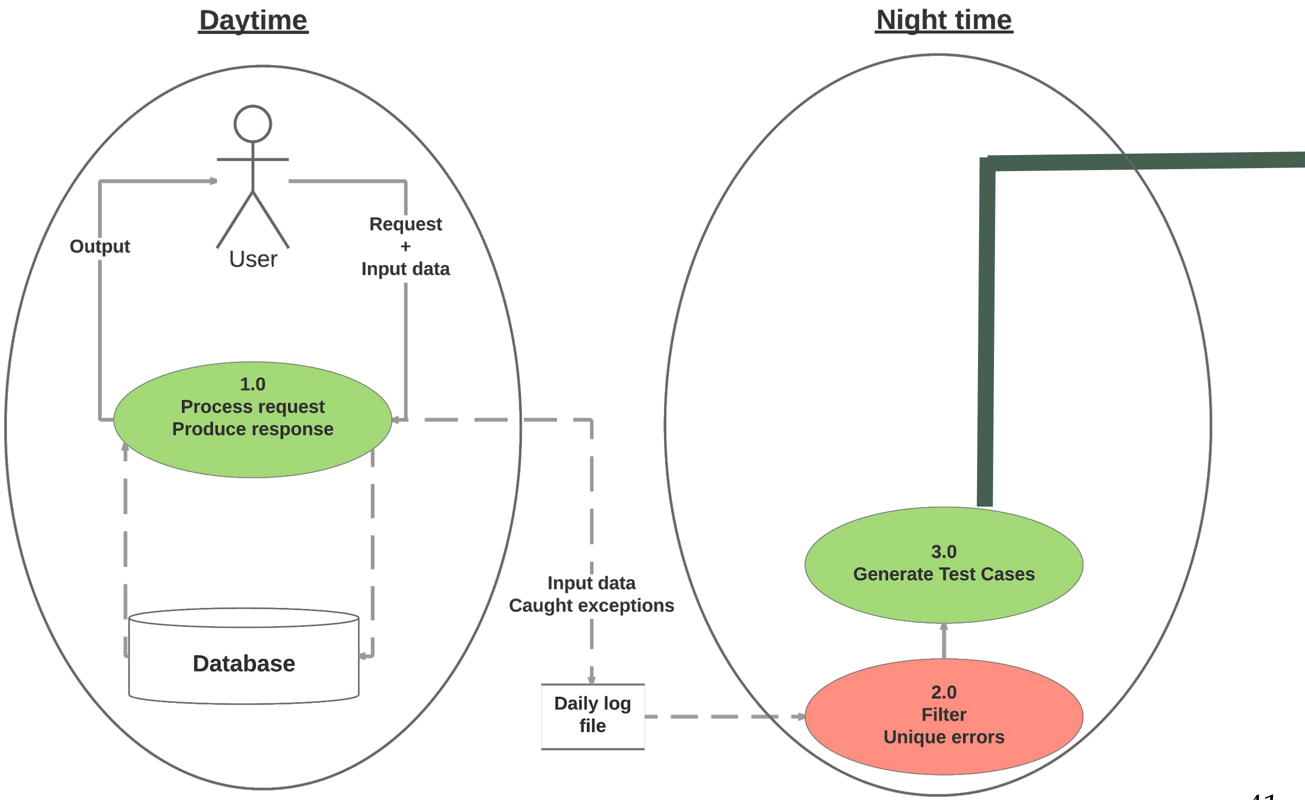
When last user logs out

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

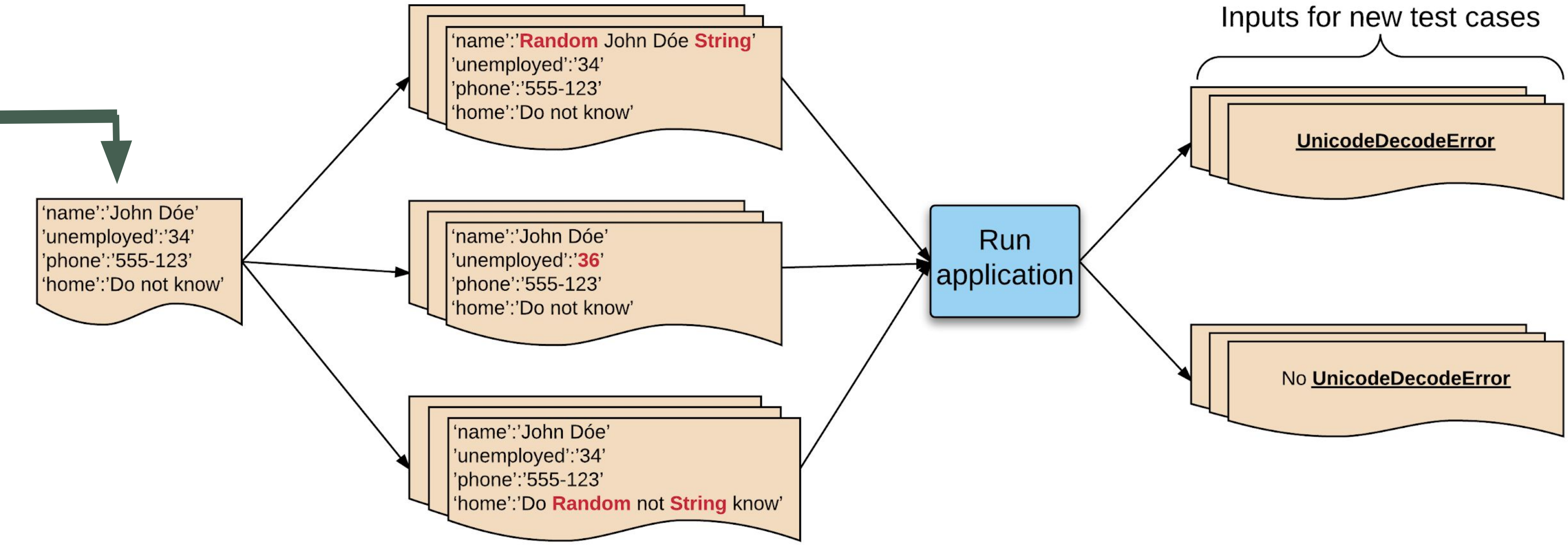


When last user logs out

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



Procedure 3.0



‘name’:’John Dóe’
’unemployed’:’34’
’phone’:’555-123’
‘home’:’Do not know’

‘name’:’**Random** John Dóe **String**’
’unemployed’:’34’
’phone’:’555-123’
‘home’:’Do not know’

‘name’:’John Dóe’
’unemployed’:’**36**’
’phone’:’555-123’
‘home’:’Do not know’

‘name’:’John Dóe’
’unemployed’:’34’
’phone’:’555-123’
‘home’:’Do **Random** not **String** know’

Run application

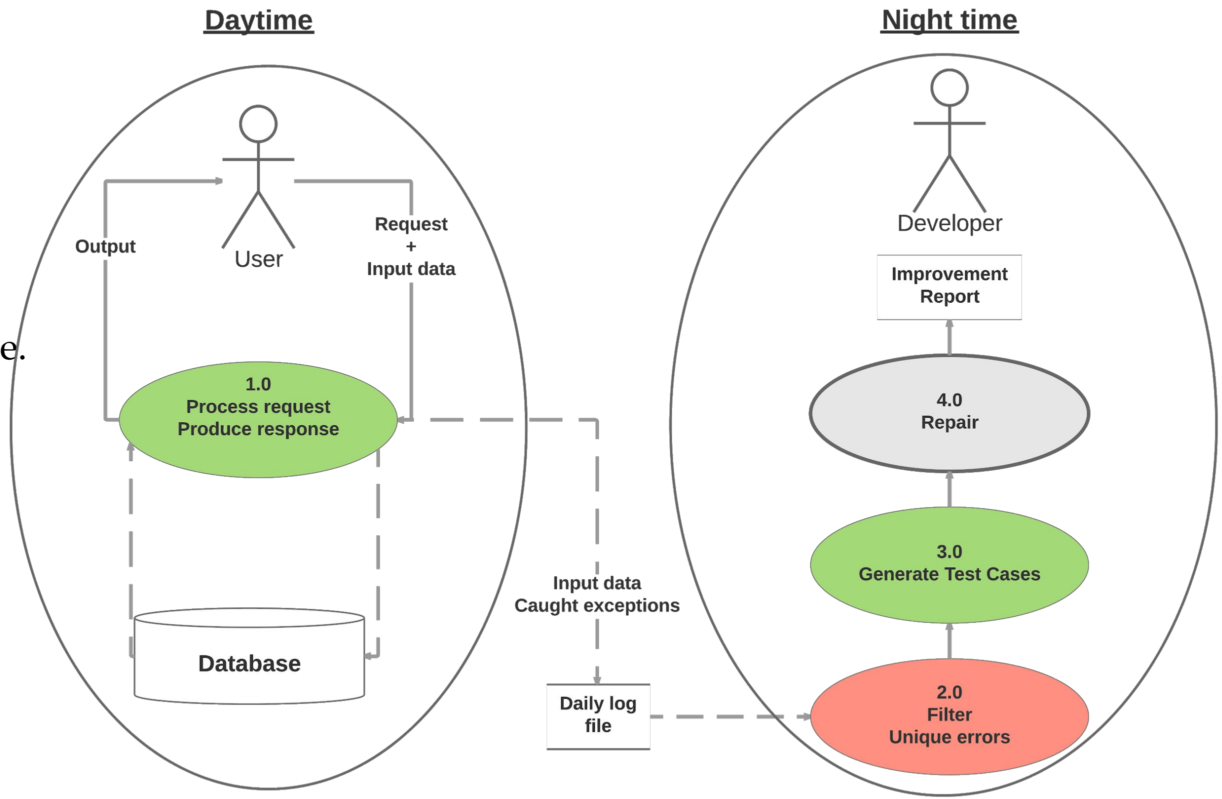
Inputs for new test cases

UnicodeDecodeError

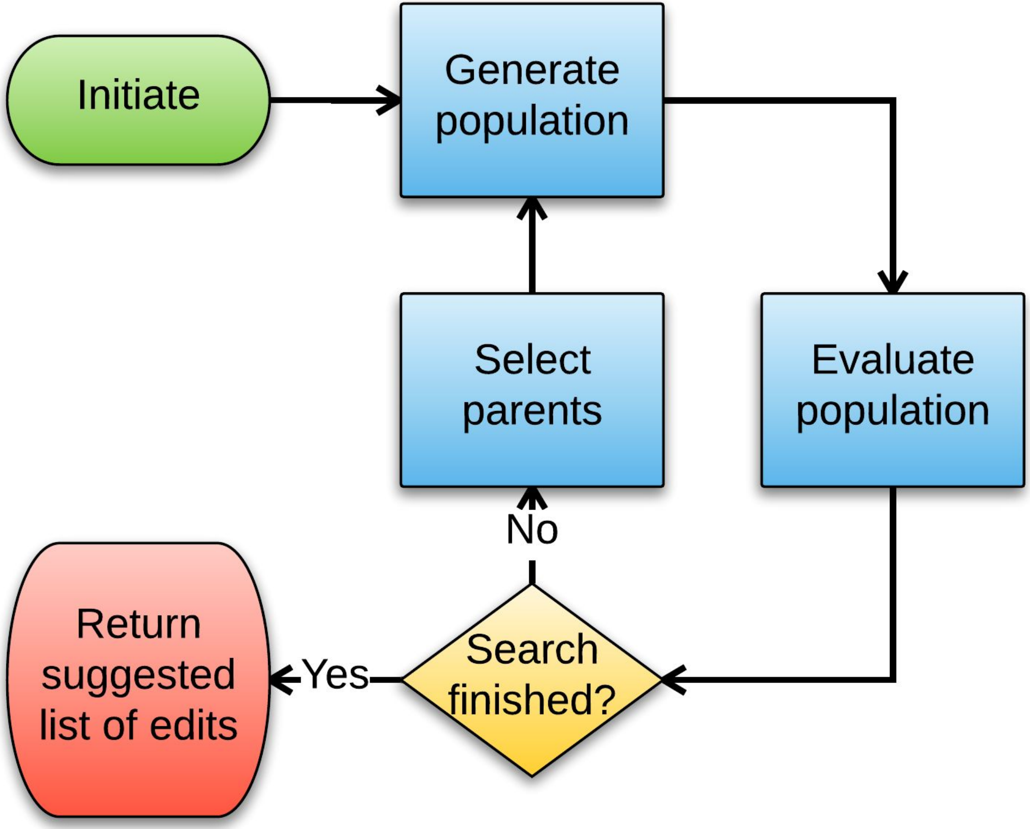
No **UnicodeDecodeError**

When last user logs out

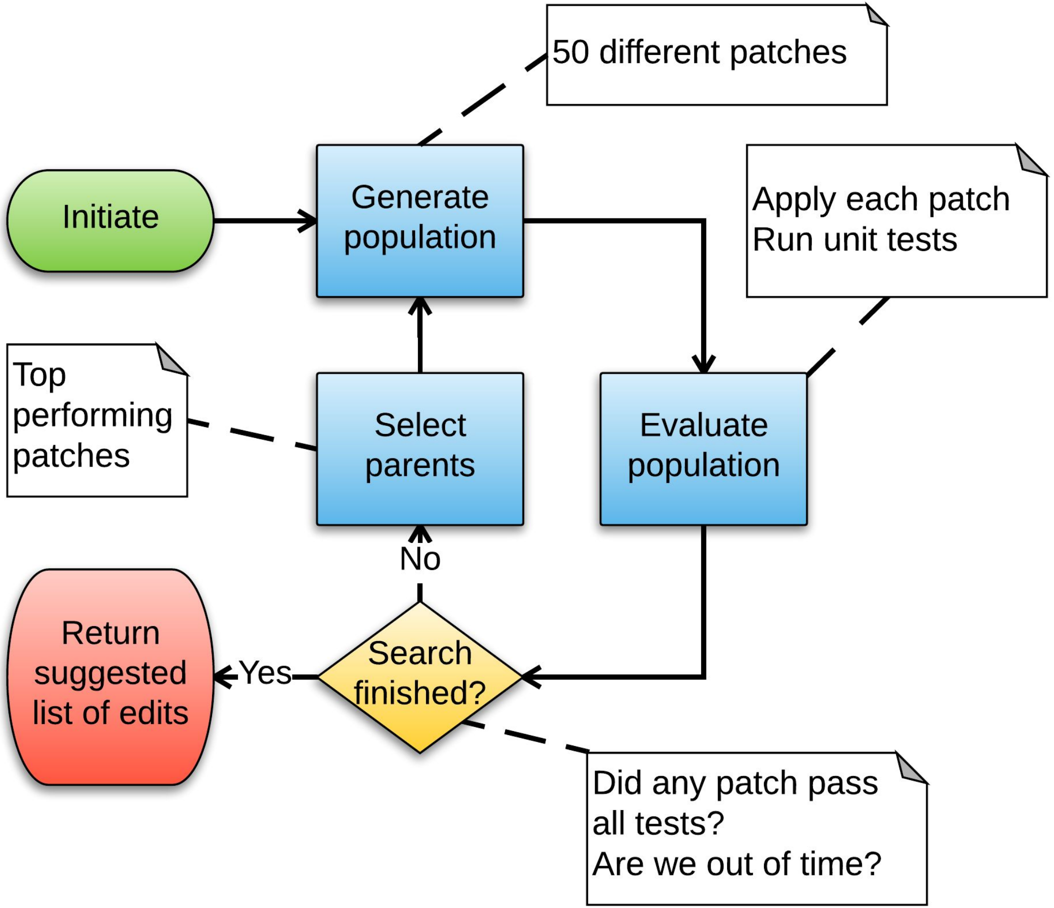
1. Procedure 2.0 started
   - Sorts and filters the day’s exceptions
2. Procedure 3.0
   - Emulates input data, type, size and structure.
   - Produces test cases
3. Procedure 4.0
   - Genetic Improvement
   - Parallel process on the server
   - Outputs report for developer



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



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



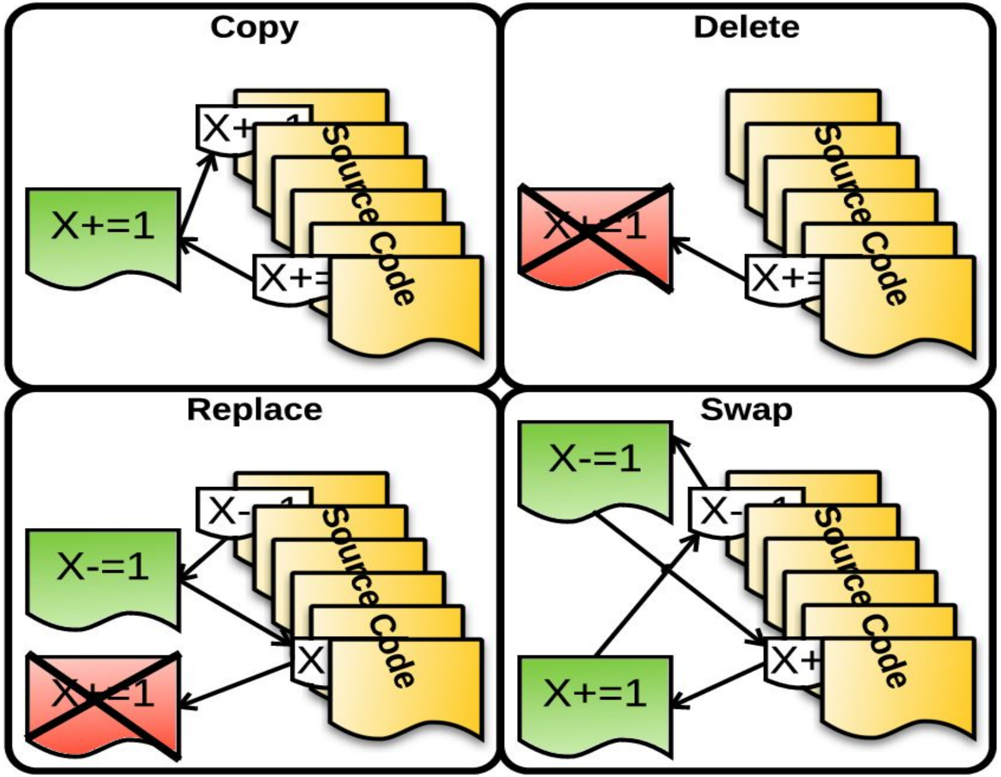
4 different types of implemented **Edits**

Primitive types:

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

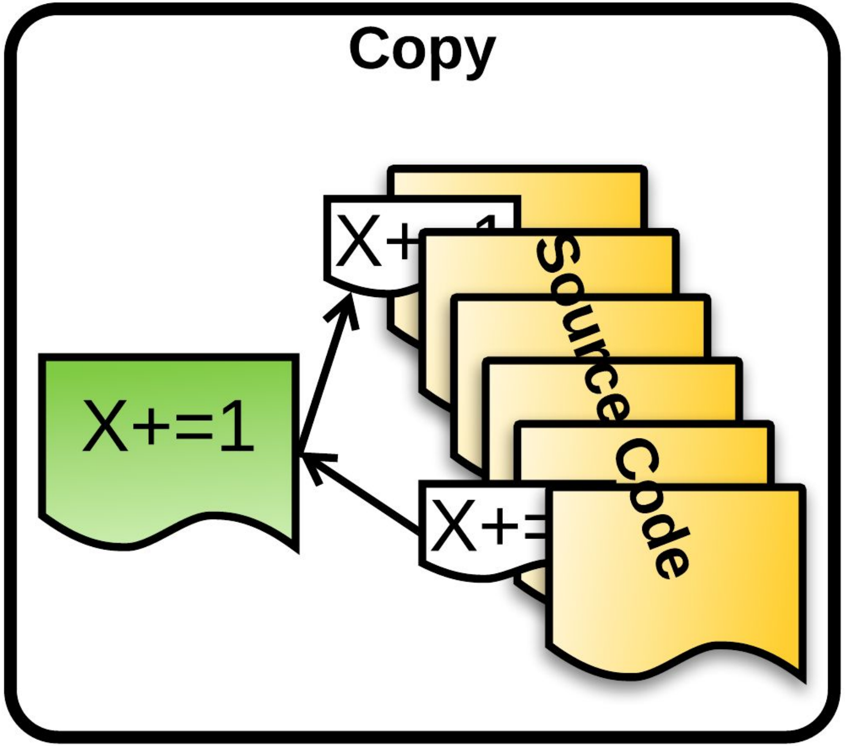
Composite types:

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



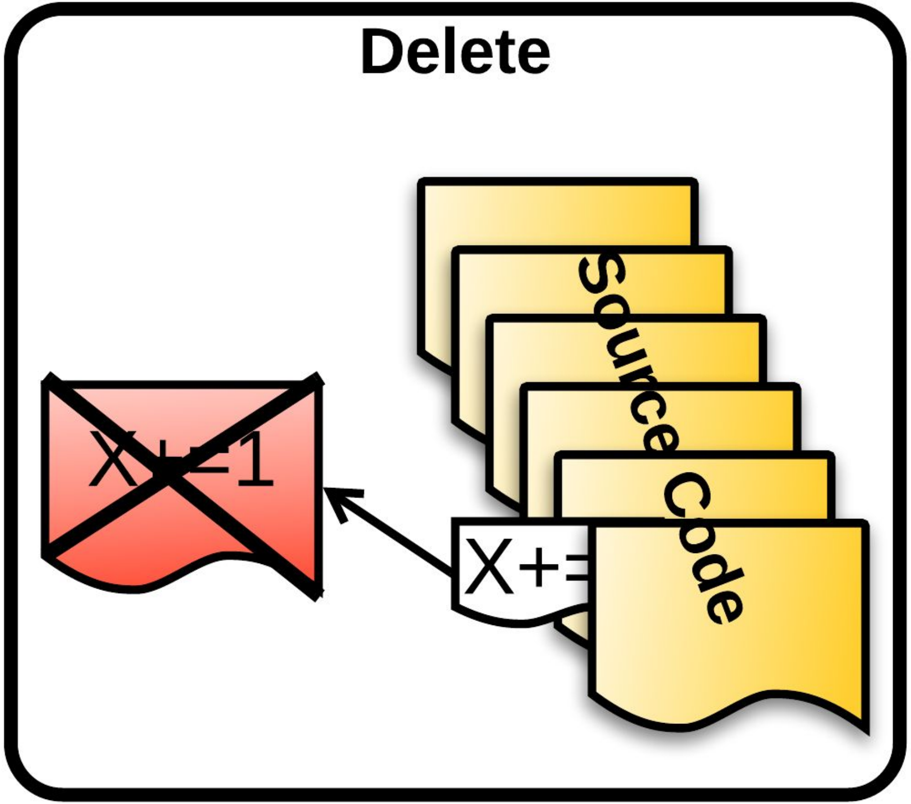
Copy

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



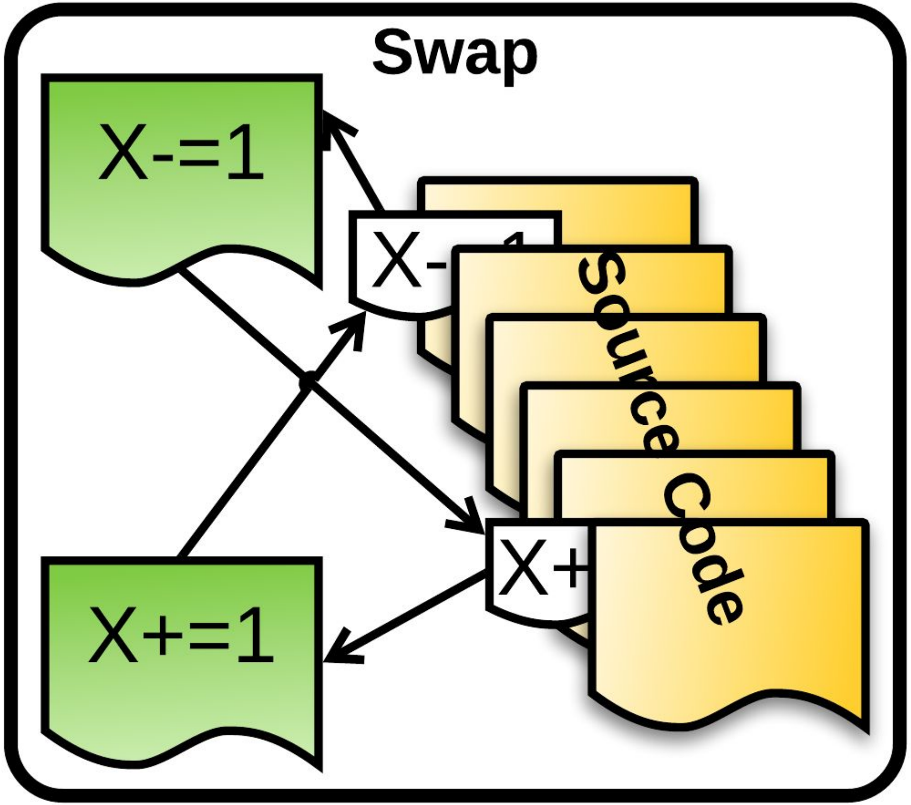
Delete

- 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"



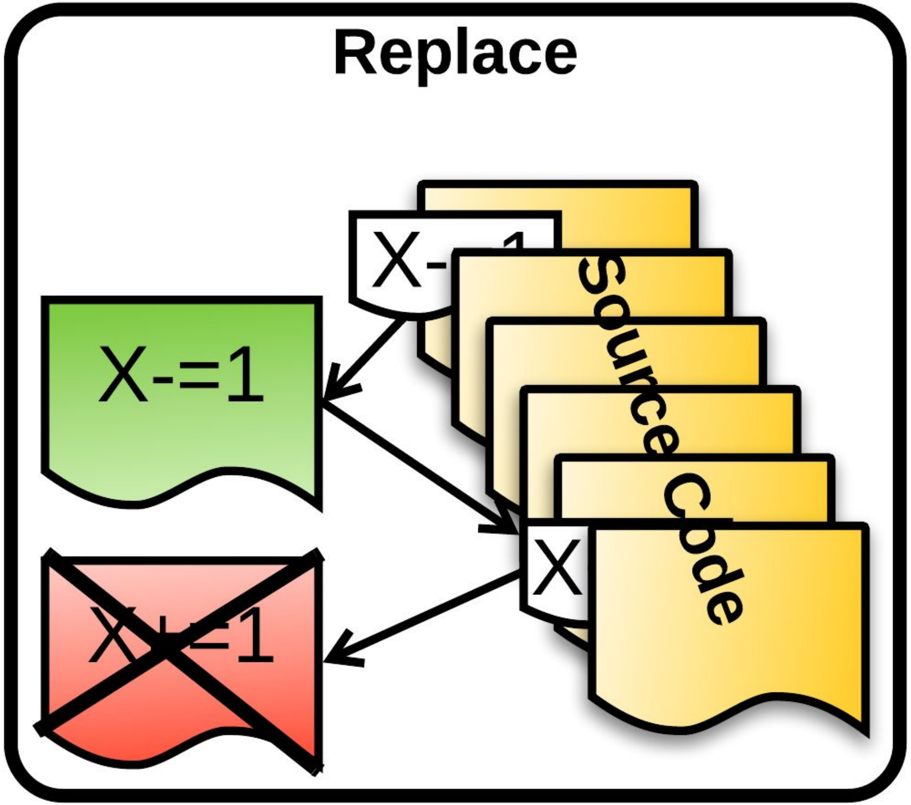
Swap

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



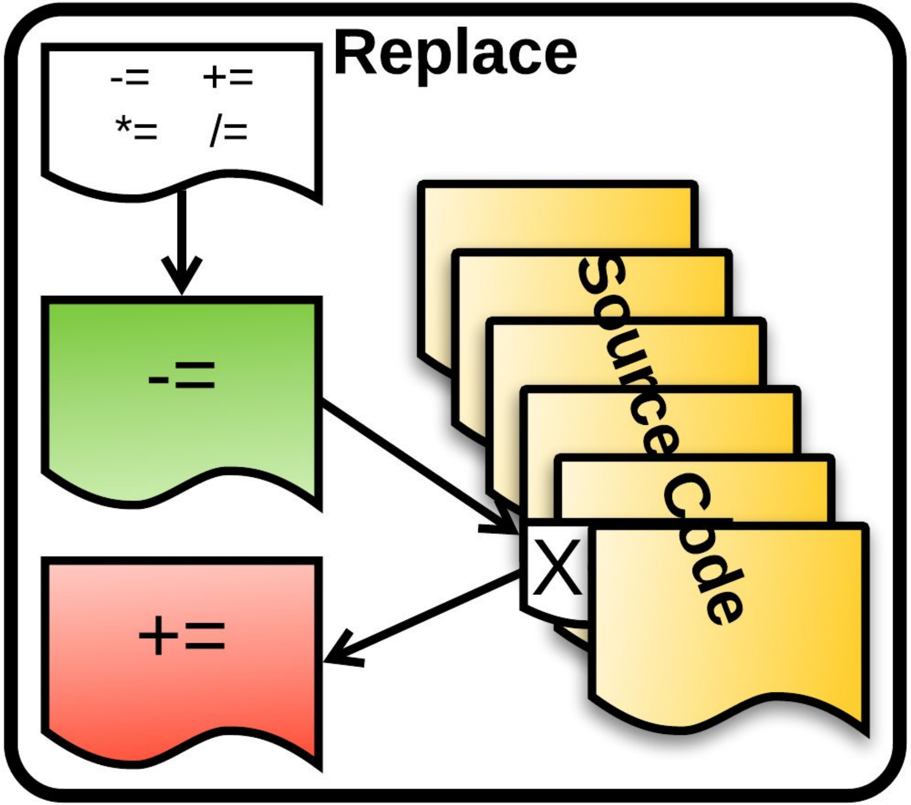
Replace

- Copies one line above another
- Then deletes that line



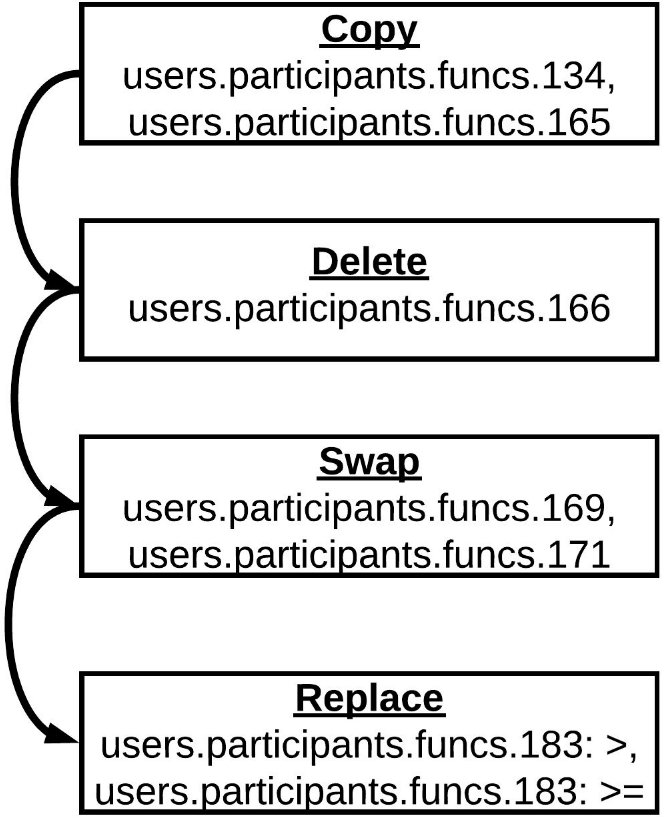
Replace -- extra

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



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



Copy
users.participants.funcs.134,
users.participants.funcs.165

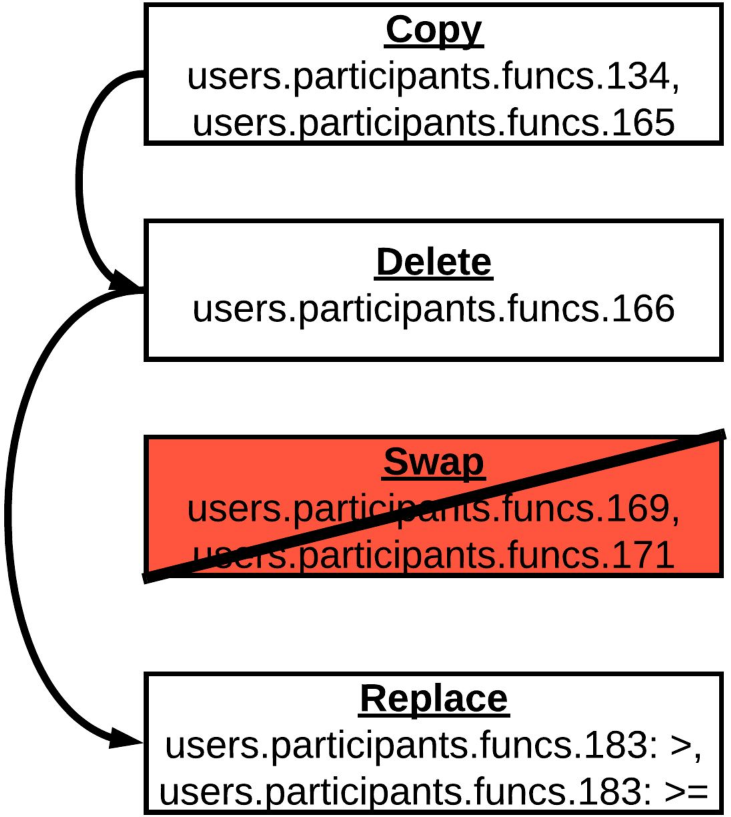
Delete
users.participants.funcs.166

Swap
users.participants.funcs.169,
users.participants.funcs.171

Replace
users.participants.funcs.183: >,
users.participants.funcs.183: >=

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



Copy
users.participants.funcs.134,
users.participants.funcs.165

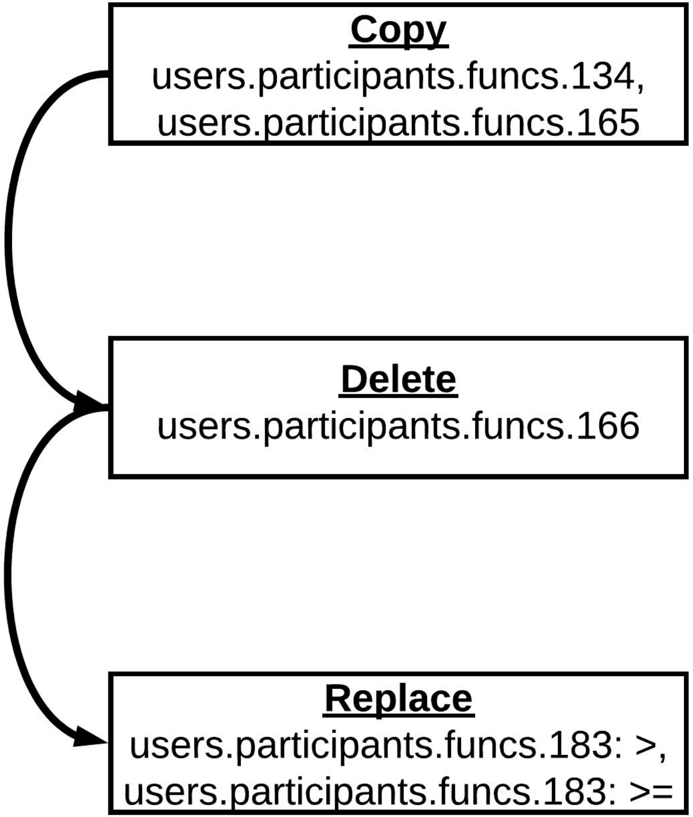
Delete
users.participants.funcs.166

~~**Swap**~~
~~users.participants.funcs.169,~~
~~users.participants.funcs.171~~

Replace
users.participants.funcs.183: >,
users.participants.funcs.183: >=

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



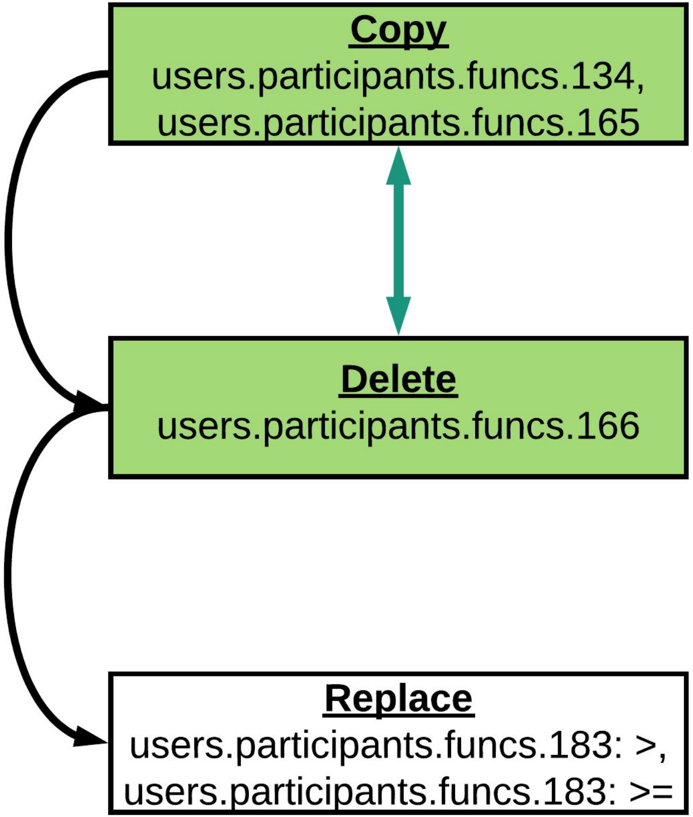
Copy
users.participants.funcs.134,
users.participants.funcs.165

Delete
users.participants.funcs.166

Replace
users.participants.funcs.183: >,
users.participants.funcs.183: >=

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



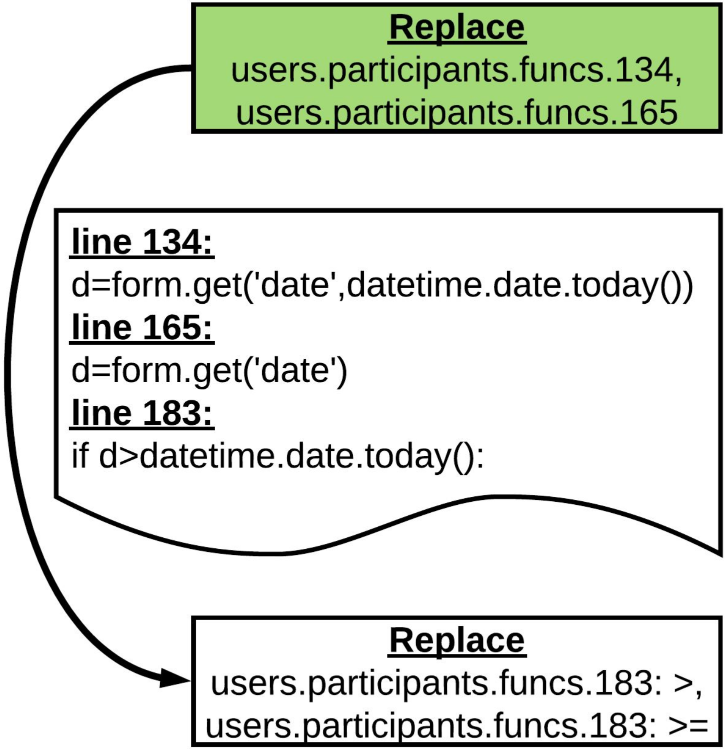
Copy
users.participants.funcs.134,
users.participants.funcs.165

Delete
users.participants.funcs.166

Replace
users.participants.funcs.183: >,
users.participants.funcs.183: >=

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



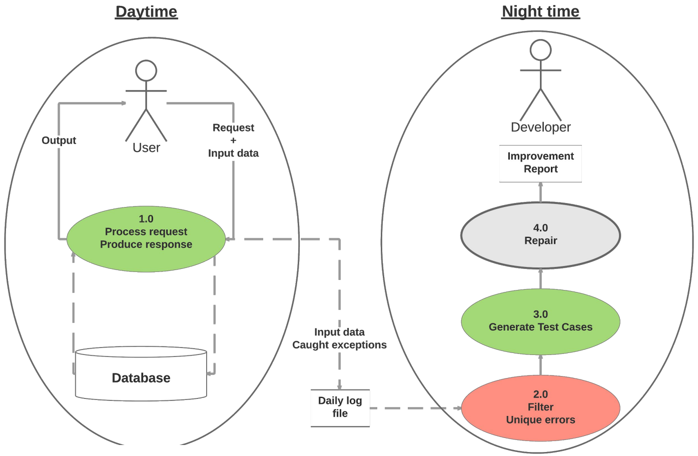
Replace
users.participants.funcs.134,
users.participants.funcs.165

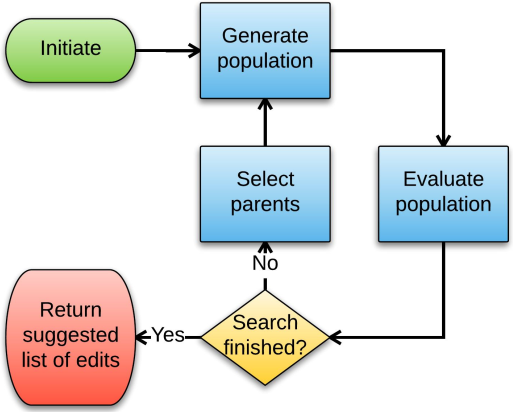
line 134:
d=form.get('date',datetime.date.today())
line 165:
d=form.get('date')
line 183:
if d>datetime.date.today():

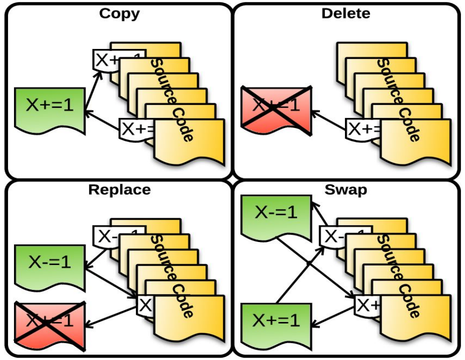
Replace
users.participants.funcs.183: >,
users.participants.funcs.183: >=

Summary

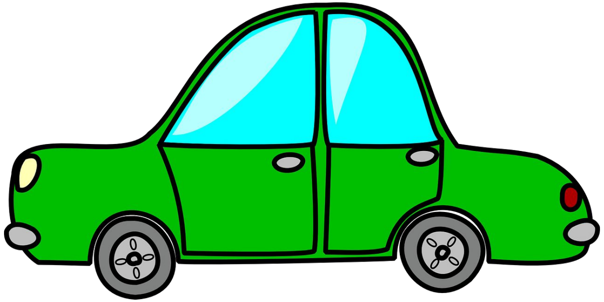
- Real-world example
- Catches inputs that produce crashes
- Line(-ish) based GI
    - 4 types of edits
- Overnight repair
- Developers are the gatekeepers







Faster



Another example of GI in action

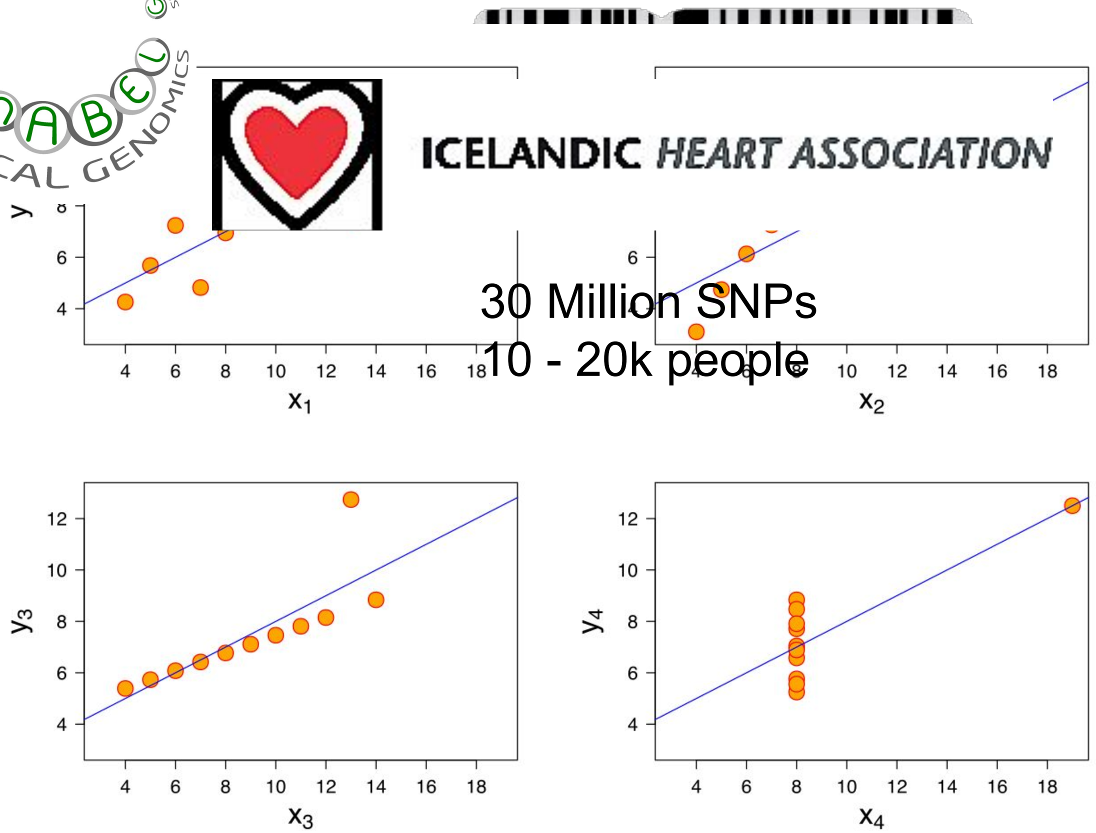
Saemundur O. Haraldsson, John R. Woodward, Alexander E. I. Brownlee, Albert V. Smith, and Vilmundur Gudnason. 2017. Genetic improvement of runtime and its fitness landscape in a bioinformatics application. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '17). ACM,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

The software

ProbABEL

- 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



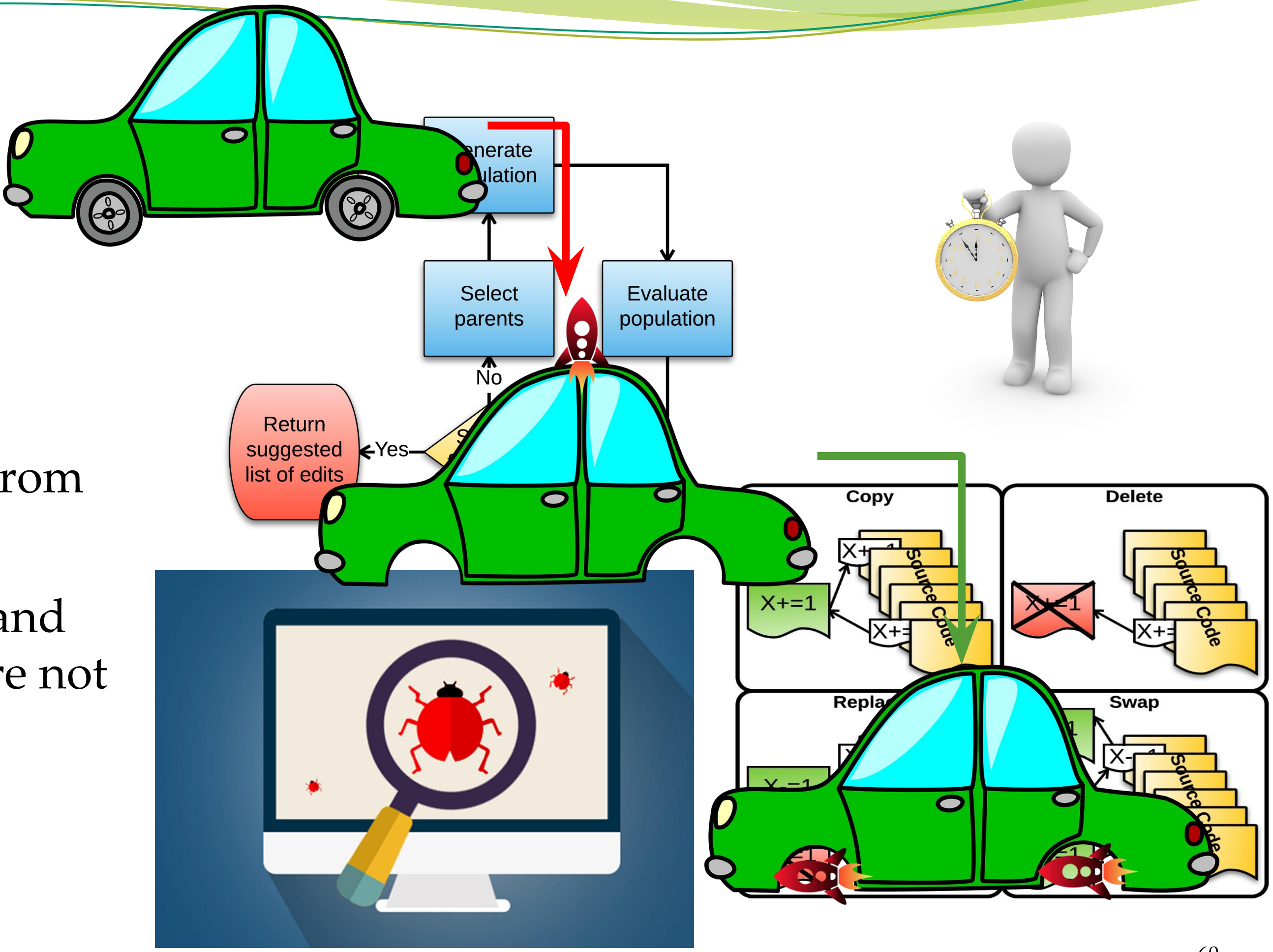
ICELANDIC HEART ASSOCIATION

30 Million SNPs
10 - 20k people

http://www.genabel.org/packages/ProbABEL

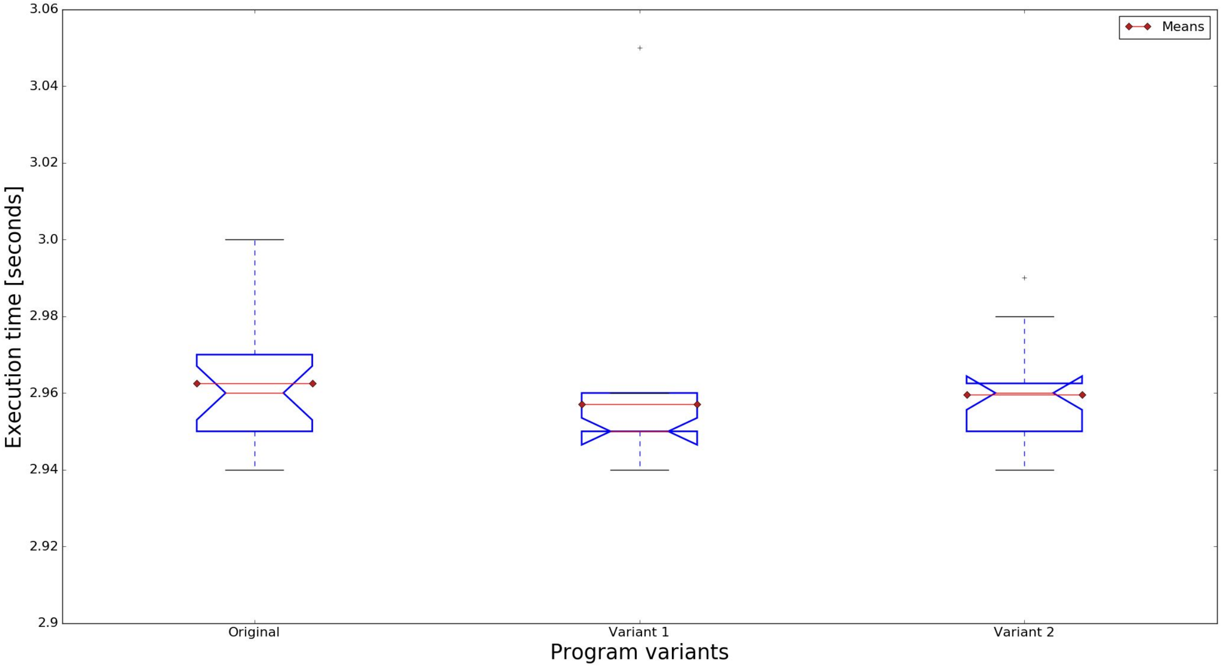
The GI setup

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



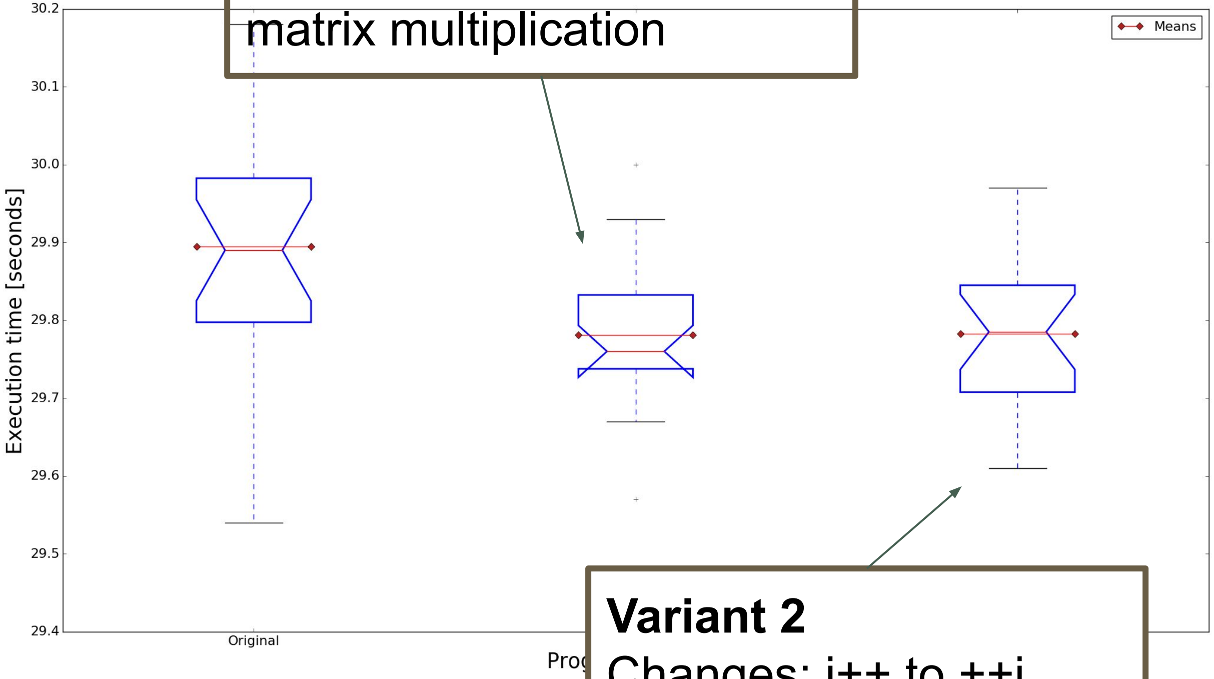
Results

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



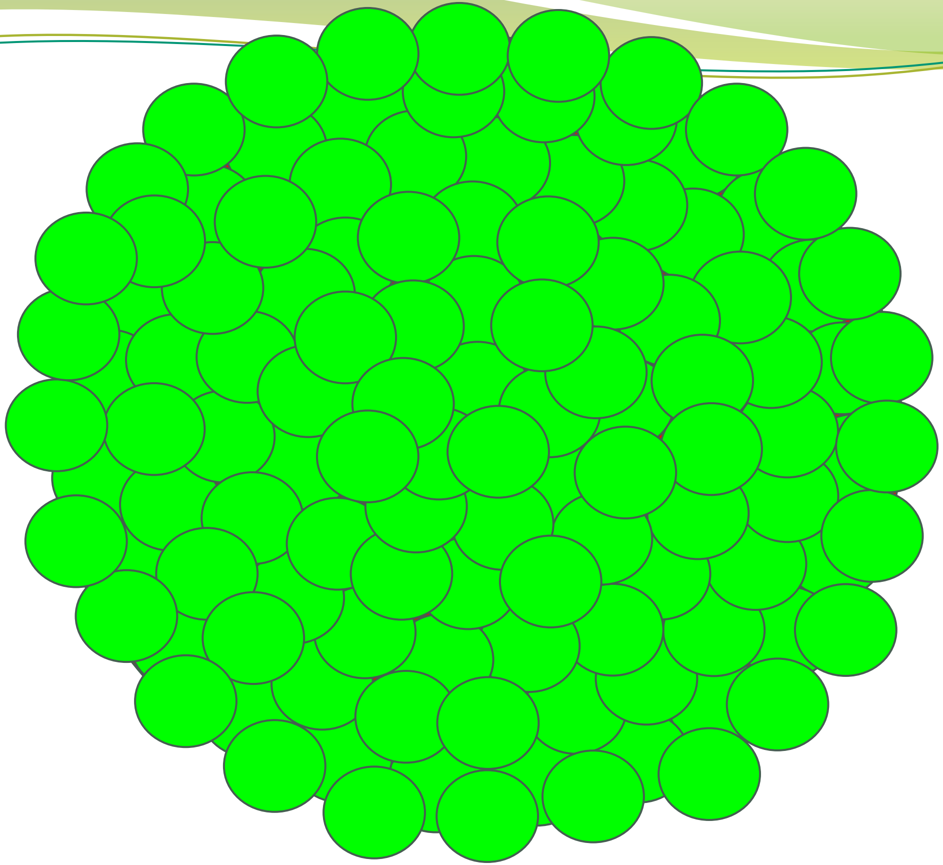
Results

- 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



Variant 1
Deletes a single line that performs an expensive matrix multiplication

Variant 2
Changes: i++ to ++i



🟢 Gained improvement per execution

Overview

- Introduction
- Example: Fixing Bugs
- Getting involved
- Summary and Q&A

Get involved with GI in No time - or GIN

GI in No Time

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

(Inaugural paper at GI@GECCO 2017)

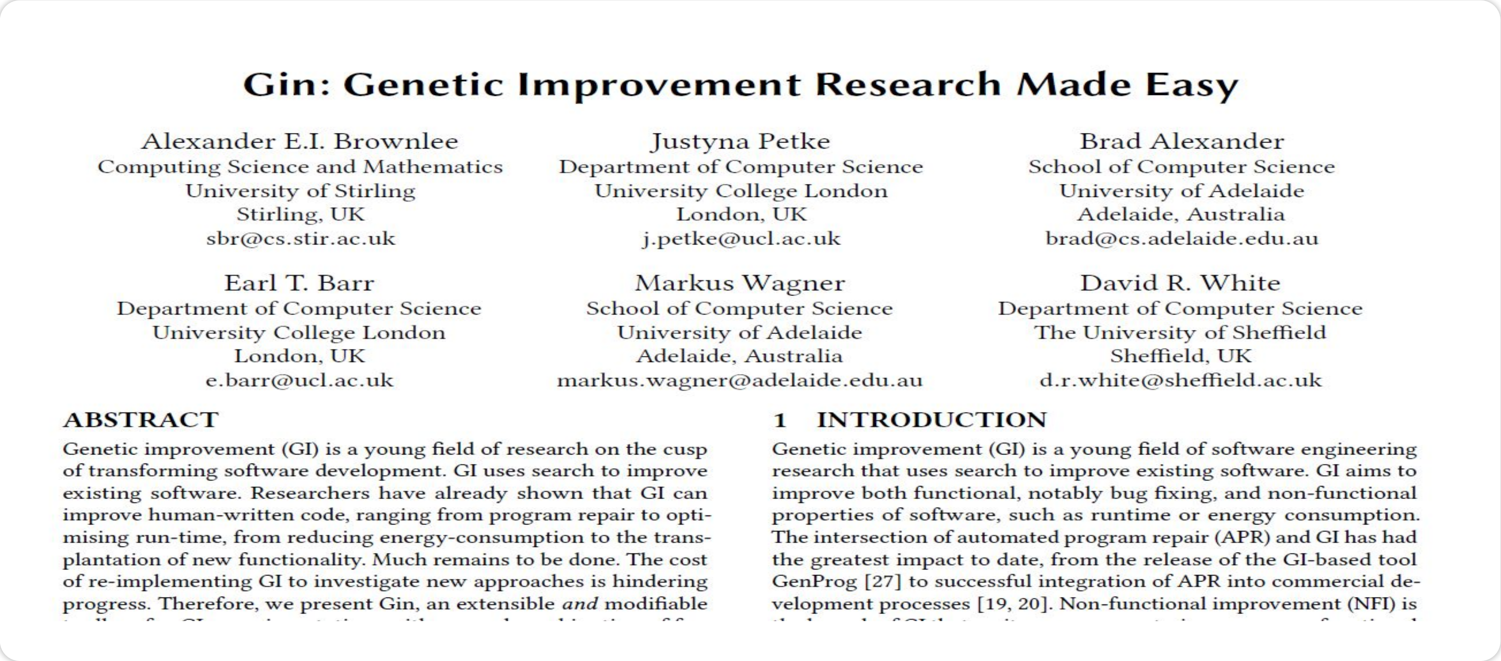
Available at
https://github.com/gintool/gin

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

v2.0 published in June 2019
"Gin: Genetic Improvement Research Made Easy" (GECCO 2019)

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
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

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 transplantation 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

Earl T. Barr

Sandy Brownlee

Justyna Petke

Markus Wagner

Also uses GIN in teaching since 2017

https://tinyurl.com/giassignment

David R. White

Genetic Improvement

- Many success stories

- …however, these typically need at GI expert in the loop

- Greater understanding needed of what GI approaches work best and where (search methods, edit types, target languages, ...)

- 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

Gin Design

Java

Maven™

Gradle Build Tool

JAVAPARSER

FOR PROCESSING JAVA CODE

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

GUAVA

⋮

Libraries

MIT

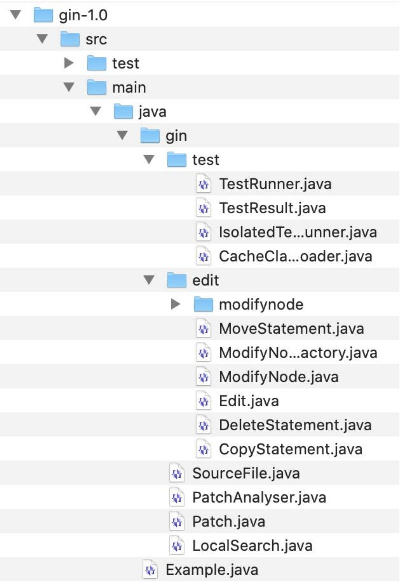
License

What's in Gin?

- Implementations of edits for source code
- Evaluate edits: compile and run JUnit tests
- Searches and Samplers
- Test generation (EvoSuite)
- Profiler to identify hot methods (hprof)
- Build tool integration (Maven, Gradle)

Let's see those in more detail...

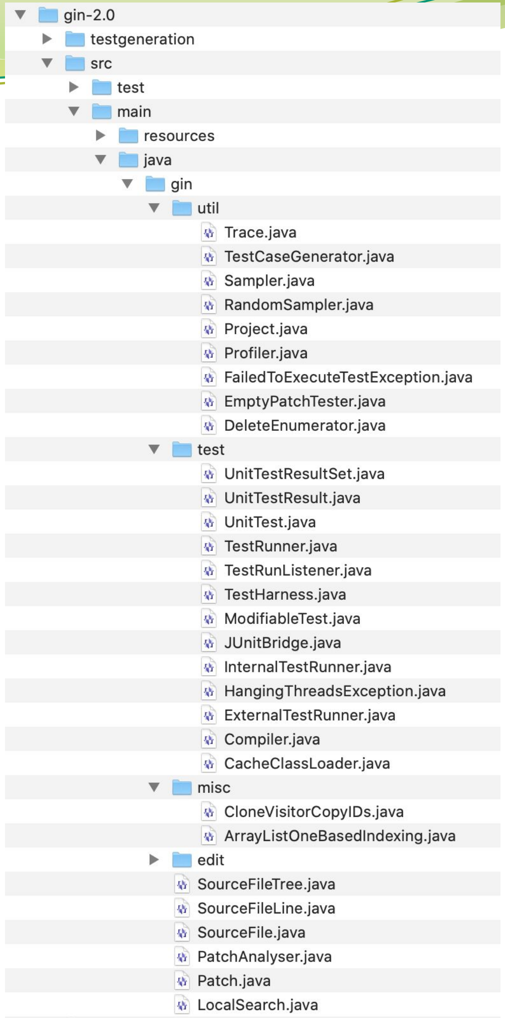
Vanilla GIN Version 1.0



```
▼ gin-1.0
  ▼ src
    ▶ test
    ▼ main
      ▼ java
        ▼ gin
          ▼ test
              TestRunner.java
              TestResult.java
              IsolatedTe...unner.java
              CacheCla...oader.java
          ▼ edit
            ▶ modifynode
              MoveStatement.java
              ModifyNo...actory.java
              ModifyNode.java
              Edit.java
              DeleteStatement.java
              CopyStatement.java
            SourceFile.java
            PatchAnalyser.java
            Patch.java
            LocalSearch.java
          Example.java
```

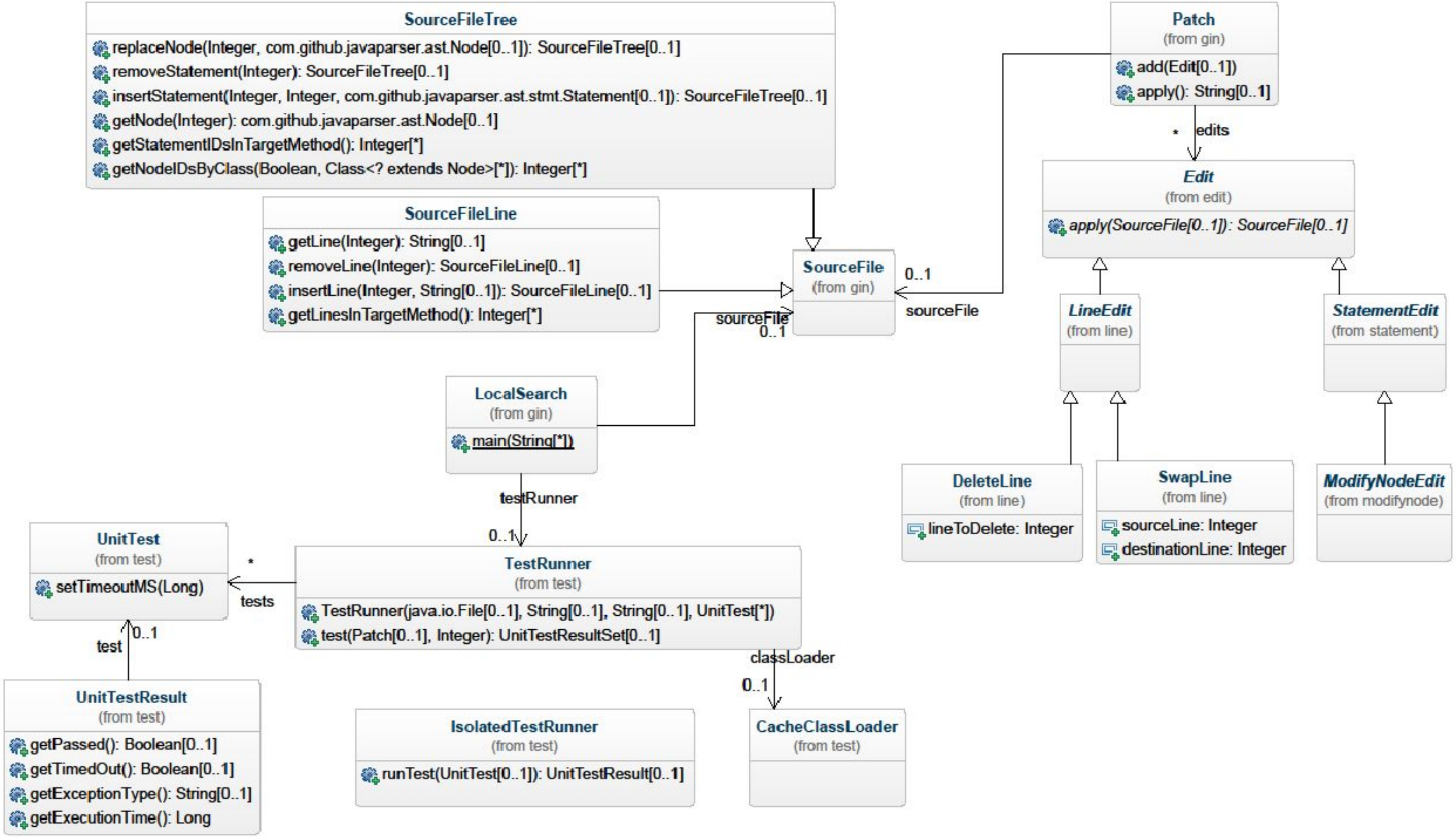
Vanilla GIN Version 2.0:

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



```
▼ gin-2.0
  ▶ testgeneration
  ▼ src
    ▶ test
    ▼ main
      ▶ resources
      ▼ java
        ▼ gin
          ▼ util
              Trace.java
              TestCaseGenerator.java
              Sampler.java
              RandomSampler.java
              Project.java
              Profiler.java
              FailedToExecuteTestException.java
              EmptyPatchTester.java
              DeleteEnumerator.java
          ▼ test
              UnitTestResultSet.java
              UnitTestResult.java
              UnitTest.java
              TestRunner.java
              TestRunListener.java
              TestHarness.java
              ModifiableTest.java
              JUnitBridge.java
              InternalTestRunner.java
              HangingThreadsException.java
              ExternalTestRunner.java
              Compiler.java
              CacheClassLoader.java
          ▼ misc
              CloneVisitorCopyIDs.java
              ArrayListOneBasedIndexing.java
          ▶ edit
            SourceFileTree.java
            SourceFileLine.java
            SourceFile.java
            PatchAnalyser.java
            Patch.java
            LocalSearch.java
```

Gin v2 Core Classes



SourceFileTree
replaceNode(Integer, com.github.javaparser.ast.Node[0..1]): SourceFileTree[0..1]
removeStatement(Integer): SourceFileTree[0..1]
insertStatement(Integer, Integer, com.github.javaparser.ast.stmt.Statement[0..1]): SourceFileTree[0..1]
getNode(Integer): com.github.javaparser.ast.Node[0..1]
getStatementIDsInTargetMethod(): Integer[*]
getNodeIDsByClass(Boolean, Class<? extends Node>[*]): Integer[*]

SourceFileLine
getLine(Integer): String[0..1]
removeLine(Integer): SourceFileLine[0..1]
insertLine(Integer, String[0..1]): SourceFileLine[0..1]
getLinesInTargetMethod(): Integer[*]

SourceFile
(from gin)

Patch
(from gin)
add(Edit[0..1])
apply(): String[0..1]

Edit
(from edit)
apply(SourceFile[0..1]): SourceFile[0..1]

LineEdit
(from line)

StatementEdit
(from statement)

DeleteLine
(from line)
lineToDelete: Integer

SwapLine
(from line)
sourceLine: Integer
destinationLine: Integer

ModifyNodeEdit
(from modifynode)

LocalSearch
(from gin)
main(String[*])

UnitTest
(from test)
setTimeoutMS(Long)

TestRunner
(from test)
TestRunner(java.io.File[0..1], String[0..1], String[0..1], UnitTest[*])
test(Patch[0..1], Integer): UnitTestResultSet[0..1]

UnitTestResult
(from test)
getPassed(): Boolean[0..1]
getTimedOut(): Boolean[0..1]
getExceptionType(): String[0..1]
getExecutionTime(): Long

IsolatedTestRunner
(from test)
runTest(UnitTest[0..1]): UnitTestResult[0..1]

CacheClassLoader
(from test)

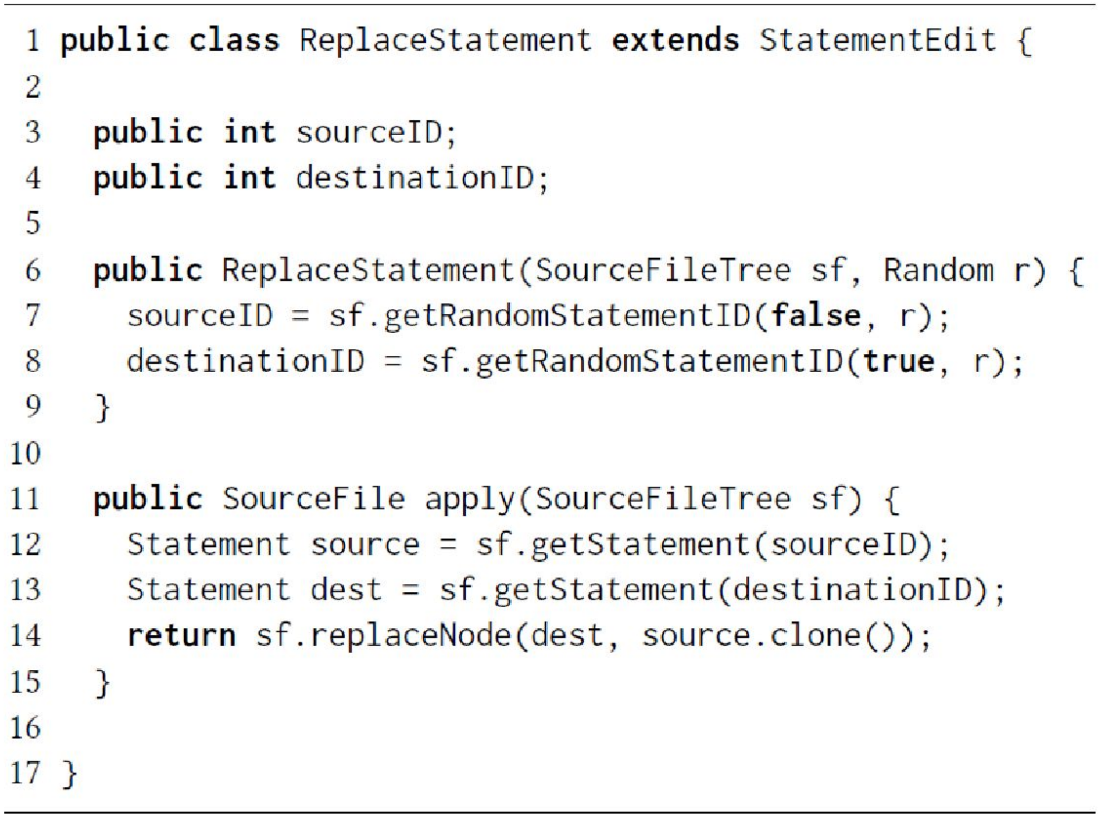
Edits

- 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)

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



```
public class ReplaceStatement extends StatementEdit {

  public int sourceID;
  public int destinationID;

  public ReplaceStatement(SourceFileTree sf, Random r) {
    sourceID = sf.getRandomStatementID(false, r);
    destinationID = sf.getRandomStatementID(true, r);
  }

  public SourceFile apply(SourceFileTree sf) {
    Statement source = sf.getStatement(sourceID);
    Statement dest = sf.getStatement(destinationID);
    return sf.replaceNode(dest, source.clone());
  }

}
```

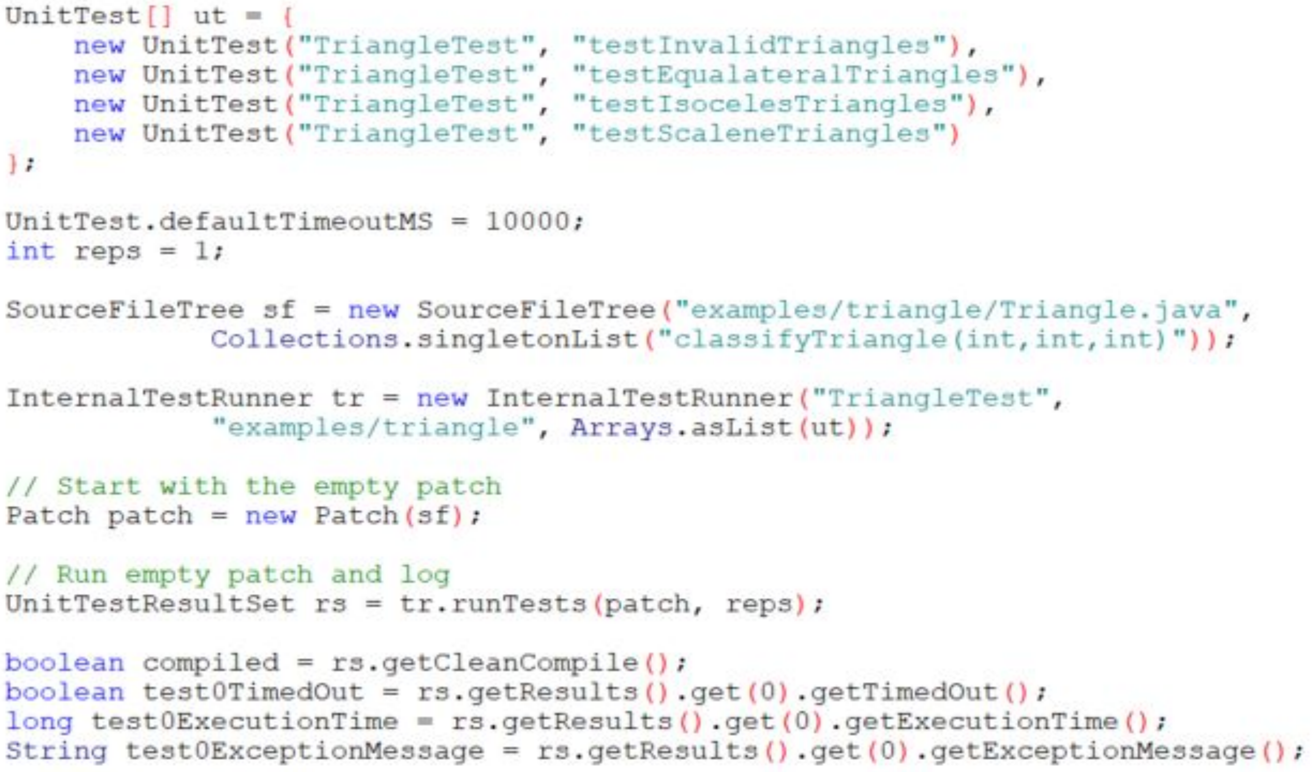
Disclaimer: this was an old version. Today, it is a little bit longer, e.g., to prevent us from replacing statements within the same parent node.

Patch Evaluation

Gin invokes test cases via Junit…

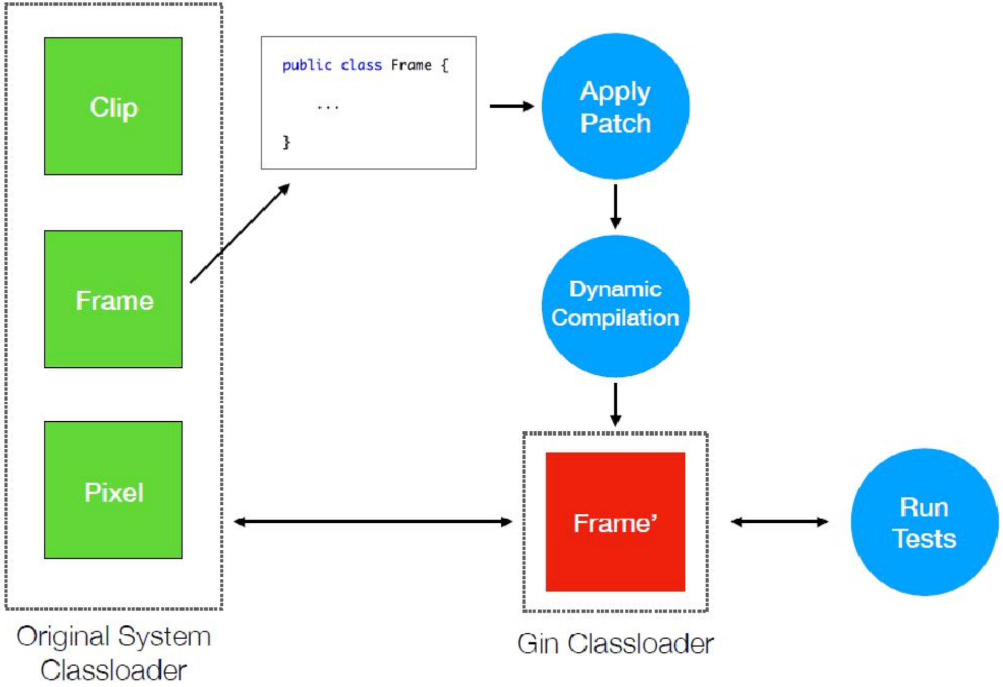
Tracks:

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



```
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();
```

Gin Compiles and Reloads on-the-fly

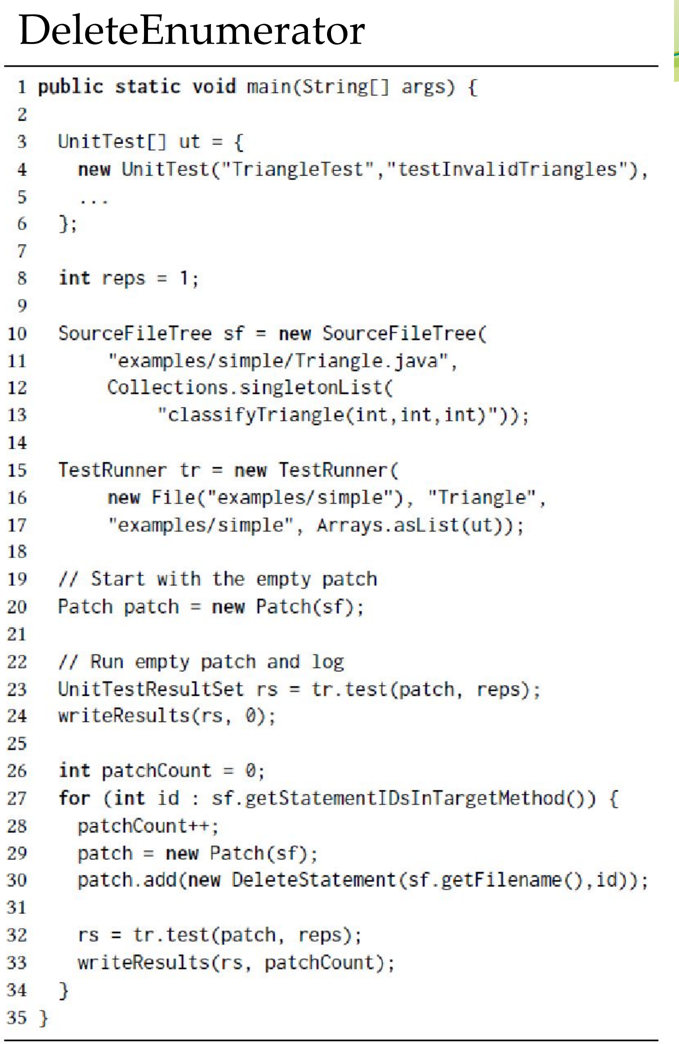


Note: If you prefer to use the more "traditional" way of writing the file to disk first - e.g., due to integration of Gin into other pipelines - then you can use a command-line flag to do so.

Sampling and Searching

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

- 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?
  - ...

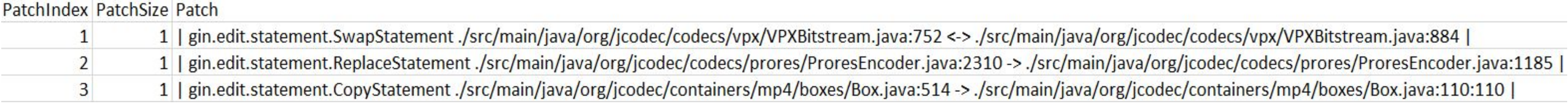


DeleteEnumerator

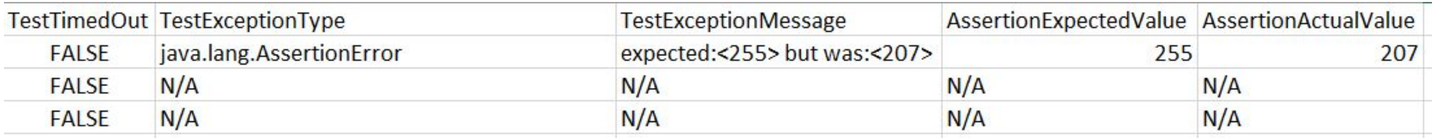
```
public static void main(String[] args) {

  UnitTest[] ut = {
    new UnitTest("TriangleTest","testInvalidTriangles"),
    ...
  };

  int reps = 1;

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

  TestRunner tr = new TestRunner(
       new File("examples/simple"), "Triangle",
       "examples/simple", Arrays.asList(ut));

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

  // Run empty patch and log
  UnitTestResultSet rs = tr.test(patch, reps);
  writeResults(rs, 0);

  int patchCount = 0;
  for (int id : sf.getStatementIDsInTargetMethod()) {
    patchCount++;
    patch = new Patch(sf);
    patch.add(new DeleteStatement(sf.getFilename(),id));

    rs = tr.test(patch, reps);
    writeResults(rs, patchCount);
  }
}
```

Sampling

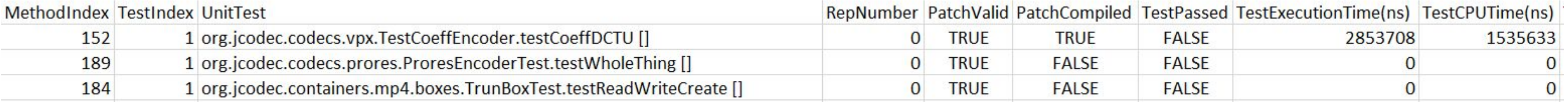
The following is one really wide output file - here of RandomSampler:



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 |

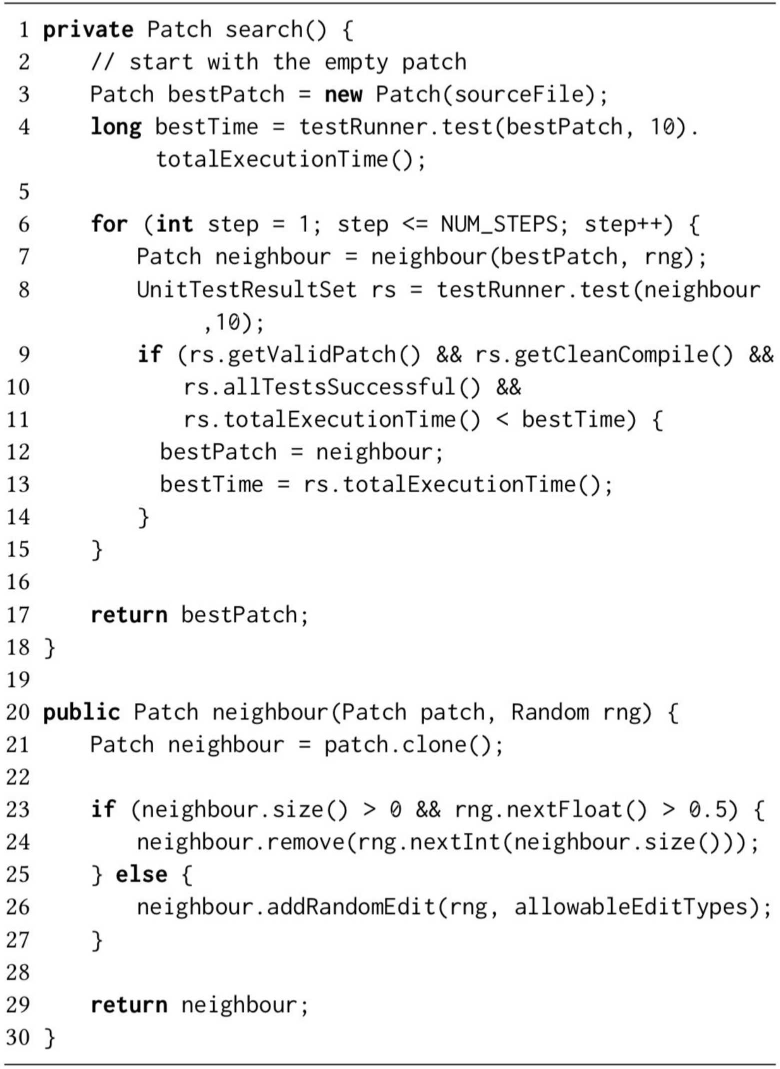


TestTimedOut	TestExceptionType	TestExceptionMessage	AssertionExpectedValue	AssertionActualValue
FALSE	java.lang.AssertionError	expected:<255> but was:<207>	255	207
FALSE	N/A	N/A	N/A	N/A
FALSE	N/A	N/A	N/A	N/A



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



```
private Patch search() {
    // start with the empty patch
    Patch bestPatch = new Patch(sourceFile);
    long bestTime = testRunner.test(bestPatch, 10).
         totalExecutionTime();

    for (int step = 1; step <= NUM_STEPS; step++) {
        Patch neighbour = neighbour(bestPatch, rng);
        UnitTestResultSet rs = testRunner.test(neighbour
             ,10);
        if (rs.getValidPatch() && rs.getCleanCompile() &&
            rs.allTestsSuccessful() &&
            rs.totalExecutionTime() < bestTime) {
          bestPatch = neighbour;
          bestTime = rs.totalExecutionTime();
        }
    }

    return bestPatch;
}

public Patch neighbour(Patch patch, Random rng) {
    Patch neighbour = patch.clone();

    if (neighbour.size() > 0 && rng.nextFloat() > 0.5) {
        neighbour.remove(rng.nextInt(neighbour.size()));
    } else {
        neighbour.addRandomEdit(rng, allowableEditTypes);
    }

    return neighbour;
}
```

Local search, output

```
-bash-4.1$ java -jar build/gin.jar gin.LocalSearch -filename examples/triangle/Triangle.java -m "classifyTriangle(int, int, int)"
```

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
```

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
```

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:28 |, 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: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)
```

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:28 |, 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: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?

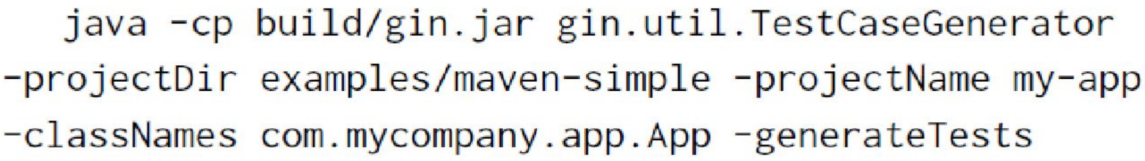
```
-bash-4.1$ cat examples/triangle/Triangle.java
public class Triangle {

          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) {

          }
          }

}
```

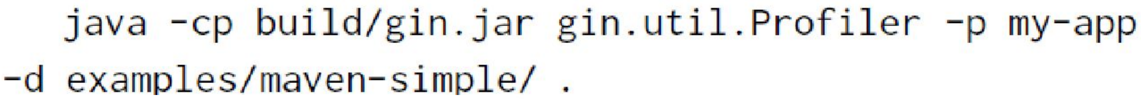
Generating tests and Profiling

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.`

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
```

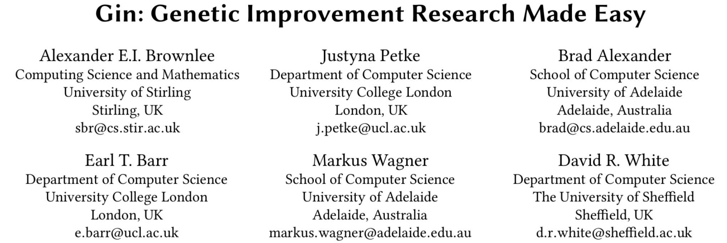
- EmptyPatchTester

```
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
```

Gin



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

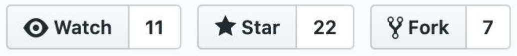
Markus Wagner
School of Computer Science
University of Adelaide
Adelaide, Australia
markus.wagner@adelaide.edu.au

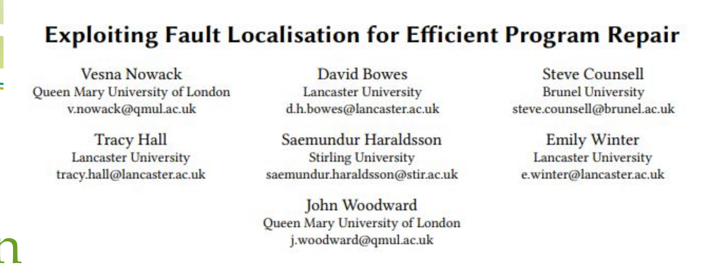
David R. White
Department of Computer Science
The University of Sheffield
Sheffield, UK
d.r.white@sheffield.ac.uk

- 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





Exploiting Fault Localisation for Efficient Program Repair

Vesna Nowack
Queen Mary University of London
v.nowack@qmul.ac.uk

David Bowes
Lancaster University
d.h.bowes@lancaster.ac.uk

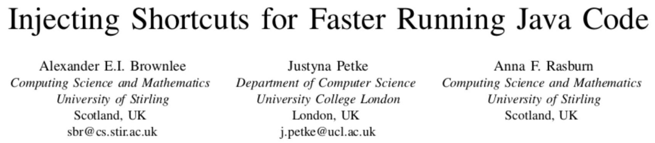
Steve Counsell
Brunel University
steve.counsell@brunel.ac.uk

Tracy Hall
Lancaster University
tracy.hall@lancaster.ac.uk

Saemundur Haraldsson
Stirling University
saemundur.haraldsson@stir.ac.uk

Emily Winter
Lancaster University
e.winter@lancaster.ac.uk

John Woodward
Queen Mary University of London
j.woodward@qmul.ac.uk

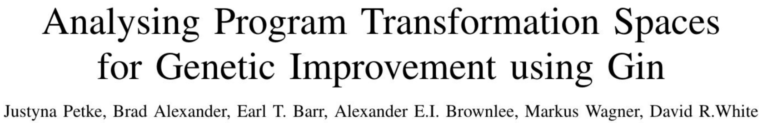


Injecting Shortcuts for Faster Running Java Code

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

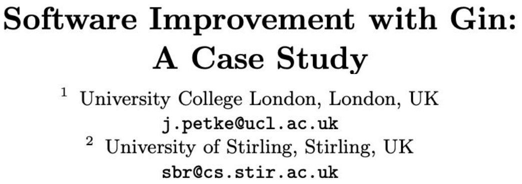
Justyna Petke
Department of Computer Science
University College London
London, UK
j.petke@ucl.ac.uk

Anna F. Rasburn
Computing Science and Mathematics
University of Stirling
Scotland, UK



Analysing Program Transformation Spaces for Genetic Improvement using Gin

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



Software Improvement with Gin: A Case Study

[1] University College London, London, UK
j.petke@ucl.ac.uk
[2] University of Stirling, Stirling, UK
sbr@cs.stir.ac.uk

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

Overview

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

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?

Saæmundur (Saemi) Haraldsson <soh@cs.stir.ac.uk>
John Woodward <j.woodward@qmul.ac.uk>
Alexander (Sandy) Brownlee <sbr@cs.stir.ac.uk>

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

S. O. Haraldsson, R. D. Brynjolfsdottir, 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. Brynjolfsdottir, 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., Brynjolfsdottir, R.D., Haraldsson, S.O., Vidar, S., Gudmundsson, E.G., Brynjolfsson, 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

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