Genetic Improvement and Magpie

Aymeric Blot

Université de Rennes

January 17, 2024 — ENS Rennes Computer Science Seminars

http://www0.cs.ucl.ac.uk/staff/a.blot/files/blot_ens_2024_slides.pdf

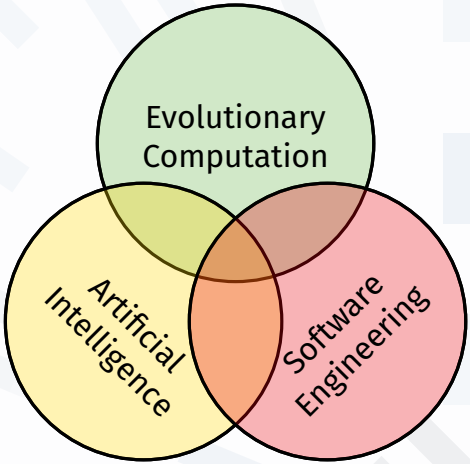
Track Record

**2011–2014: Magistère & **
- ENS Rennes
- (L3, M2) Inria Lille
- (M1) Shinshu University

2014–2018: Predoc & PhD
- University of British Colombia
- Université de Lille

2018–2022: Postdoc
- University College London

2022–present: ATER & MCF
- Université du Littoral Côte d'Opale
- Université de Rennes

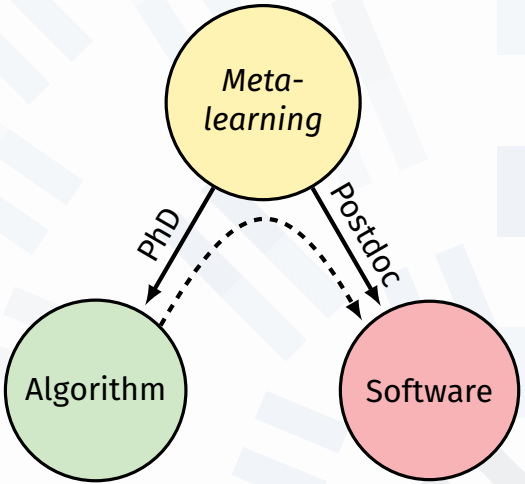


Research Experience

Evolutionary Computation
- metaheuristics (LS, GP)
- single and multi-criteria combinatorial optimisation

Automated Algorithm Design
- algorithm configuration
- parameter control

Genetic Improvement
- automated bug fixing
- non-functional properties

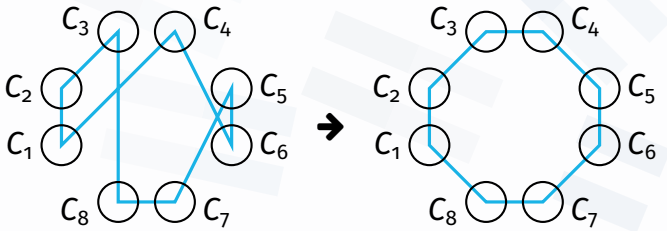


Combinatorial Optimisation

Definition

- finding the optimal solution from a finite set
- in a context in which the search space grows exponentially
- with regard to a given fitness function and potential constraints

Example: Travelling Salesman Problem (TSP)



Terminology in the Face of Combinatorial Explosion

Exact algorithms

- find provably optimal solutions, but usually don't scale
- known to be time expensive on large or difficult problems
- ➔ precision at the cost of computational efficiency

Heuristics

- strategies to explore promising regions of the search space
- are able to find good solutions very quickly
- ➔ speed and adaptability with no optimality guaranties

***Meta*heuristics**

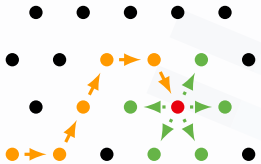
- generic high-level heuristics suitable on a large range of problems
- trajectory-based vs nature-inspired algorithms

Metaheuristic Search Algorithms

Local search

"Proximate optimality principle"

- single solution trajectory
- mutation-only



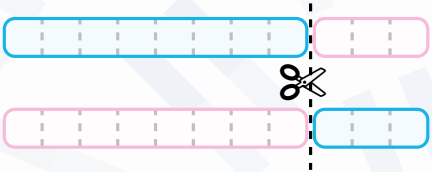
Examples

- iterated local search
- simulated annealing
- tabu search

Evolutionary computation

"Survival of the fittest"

- population of solutions
- mutation and crossover



Examples

- genetic algorithm (fixed-size solution)
- genetic programming (variable-size)

And many, *many*, `many` others...

Meta-Learning

Key idea

Searching for solutions ➔ optimising the search algorithm

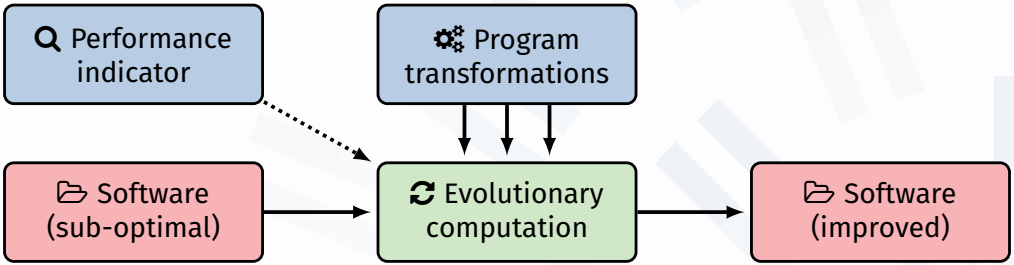
Motivations

- take into account unique ad-hoc requirements
- learn from expected characteristics of optimal solutions
- specialise strategies to given input data

Examples

- tuning hyperparameters
- selecting, combining, evolving algorithms
- designing novel procedures from scratch

Genetic Improvement (GI) of Software



Idea: evolve software (e.g., source code) to improve performance
Motivation: unknown defects, changes in specification, code rot, ...

Software Properties

Functional properties

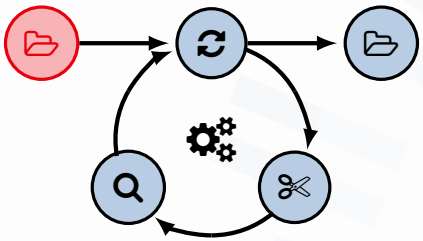
- automated program repair
- feature transplantation
- code translation
- parallelisation
- ...

Non-functional properties

- execution time
- energy consumption
- memory usage
- solution quality
- code size
- ...

Petke et al., IEEE Transactions on Evolutionary Computation, 2018

Evolving Software in Practice



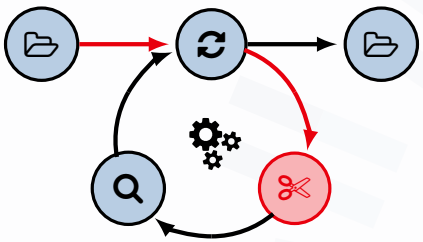
Existing software

- lines of code
- abstract syntax tree
- configuration

Initial solution

- empty list of *edits*

Evolving Software in Practice



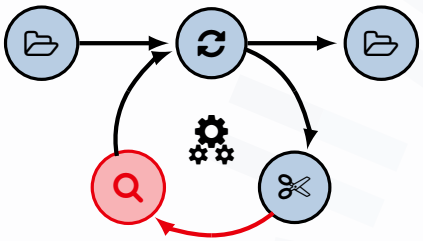
Edits

- [e.g., line] deletion
- [e.g., line] insertion
- [e.g., line] replacement

Exploration

- add new edit
- remove existing edit
- crossover

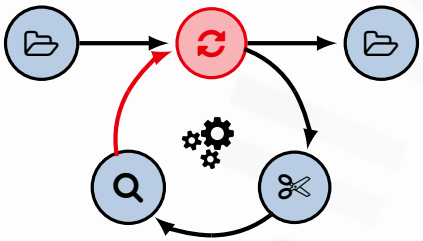
Evolving Software in Practice



Evaluation

- apply edits
- recompile
- validate
- run → *fitness*

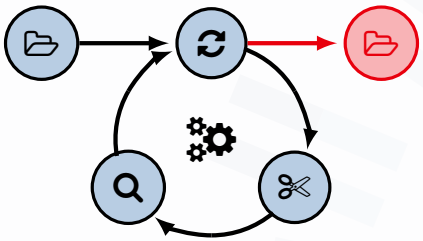
Evolving Software in Practice



Accumulate edits

- *genetic programming* (GP)
  - population
  - crossover
  - mutation
- *local search* (LS)
  - single current solution
  - mutation only

Evolving Software in Practice



Check generalisation
- minimise patch
- on new data

Manual code review
- assess semantics
- accept / patch / reject

Some Results

MiniSAT (C++)
- change of *restart* strategy (61% faster)
- disable non-essential search optimisation (22% faster)

Sat4J (Java)
- change of *restart* strategy (84% faster)
- disable learnt clause history (26% faster)

OptiPNG (C)
- remove bespoke zlib configuration (20% faster)

MOEA/D, NSGA-II (C++)
- remove unnecessary fitness computation (7% and 12% faster)
- various interesting algorithmic changes

Patch Example 1/3: MiniSAT

Inhibiting restarts: −61% CPU instructions

```
--- after: core/Solver.cc
     }else{
          // NO CONFLICT
-         if (nof_conflicts >= 0 && conflictC >= nof_conflicts || !
-             // Reached bound on number of conflicts:
-             progress_estimate = progressEstimate();
-             cancelUntil(0);
-             return l_Undef; }

          // Simplify the set of problem clauses:
          if (decisionLevel() == 0 && !simplify())
```

http://www0.cs.ucl.ac.uk/staff/a.blot/files/blot_tevc_2021_supp.pdf

Patch Example 2/3: OptiPNG

Removing bespoke configuration: −41% CPU instructions

```
--- after: src/optipng/optim.c
    png_set_compression_mem_level(write_ptr, memory_level);
    png_set_compression_strategy(write_ptr, compression_strategy);
    png_set_filter(write_ptr, PNG_FILTER_TYPE_BASE, filter_table[fi
-   if (compression_strategy != Z_HUFFMAN_ONLY &&
-       compression_strategy != Z_RLE) {
-       if (options.window_bits > 0)
-           png_set_compression_window_bits(write_ptr, options.wind
-   } else {
- #ifdef WBITS_8_OK
-       png_set_compression_window_bits(write_ptr, 8);
- #else
-       png_set_compression_window_bits(write_ptr, 9);
- #endif
-   }
```

Note: ideally a replacement, not just a deletion

https://sourceforge.net/p/optipng/patches/10/

Patch Example 3/3: MOEA/D

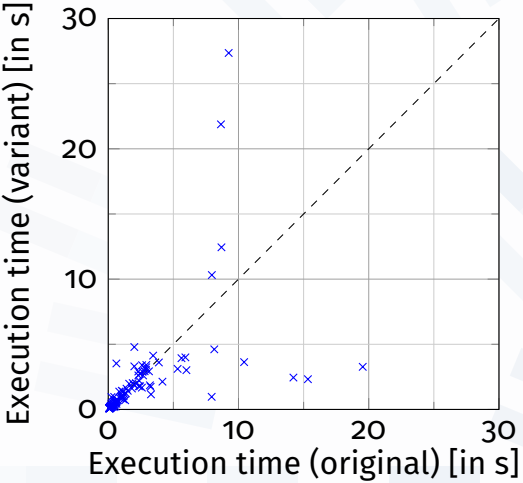
Removing IGD computation: −12% CPU instructions

```
--- after: DMOEA/dmoeafunc.h
 void CMOEAD::calc_distance() {
      distance = 0;
-     for(int i=0; i<ps.size(); i++) {
-         double min_d = 1.0e+10;
-         for(int j=0; j<population.size(); j++) {
-             double d = dist_vector(ps[i].y_obj,
-                                    population[j].indiv.y_obj);
-             if (d<min_d) min_d = d;
-         }
-         distance += min_d;
-     }
      distance /= ps.size();
 }
```

Note: final population was captured and externally reassessed

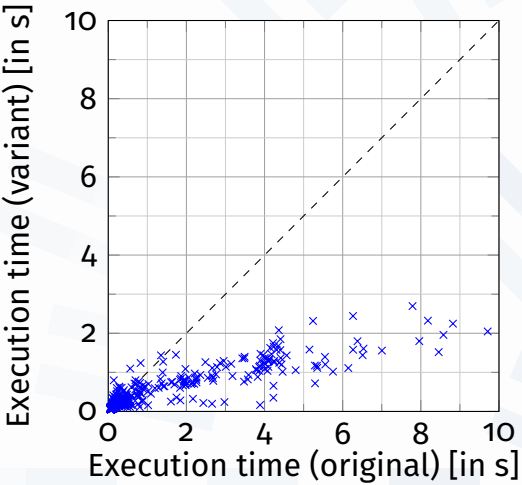
Types of Improvements 1/3: Unclear

MiniSAT: −22% CPU instructions



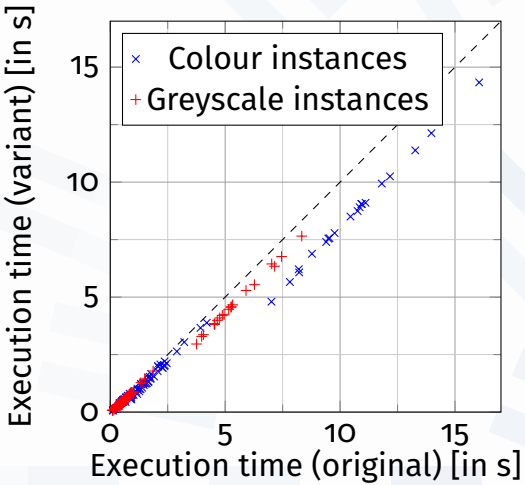
Types of Improvements 2/3: Linear

MiniSAT: −68% CPU instructions



Types of Improvements 3/3: Constant

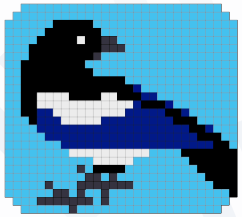
OptiPNG: −41% CPU instructions



Machine Automated General Perf. Improv. via Evolution
(of software)

Magpie

- Python framework for GI/MT/AC/CO
- for scientists!
- for end-users!



https://github.com/bloa/magpie

Magpie for End-Users

Easy to play with

- fully scenario-based
- plug and play: no dependence, no installation
- no Python knowledge *required*

Multi-paradigm

- genetic improvement
- mutation testing
- algorithm configuration (at run and compile time)

Full of goodies

- 4 types of software representations
- 4 main types of fitness functions
- 2 main types of search algorithms (LS+GP), many utilities

Magpie for Scientists

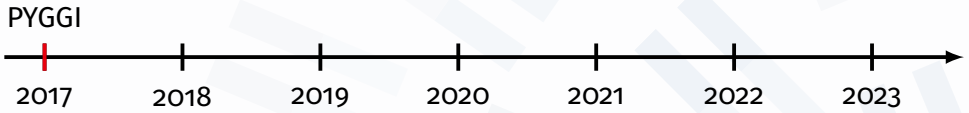
Easy to code with

- single local self-contained folder
- no installation, no compilation
- free and open-source

Tiny codebase

- 40 Python files, 3.5k loc
  - core: 14 files, 1.5k loc
  - algos: 5 files, 700 loc
  - models: 19 files, 1.3k loc
- designed to be as extensible as possible

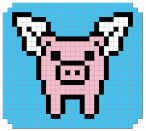
A Brief History

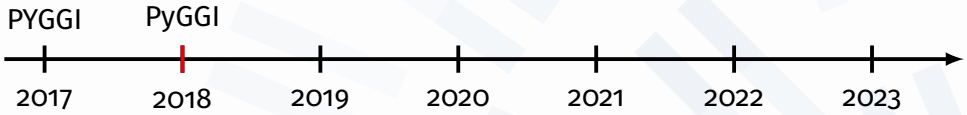


2017: PYGGI (KSC)

- lines of code (deletion, insertion, replacement)
- local search
- program repair, execution time

A Brief History

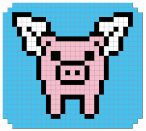


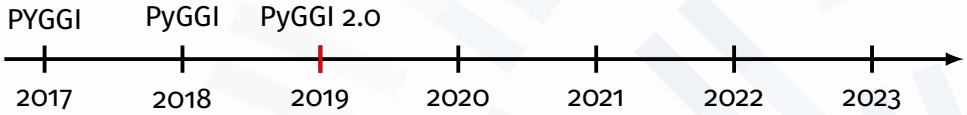


2018: PyGGI (GI@GECCO)

- Python statement AST (deletion, insertion, replacement)

A Brief History

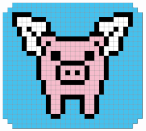


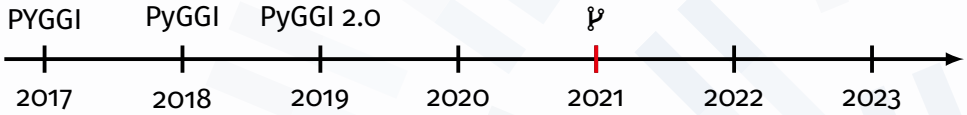


2019: PyGGI 2.0 (ESEC/FSE)

- XML/SrcML files (deletion, insertion, replacement)

A Brief History



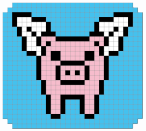


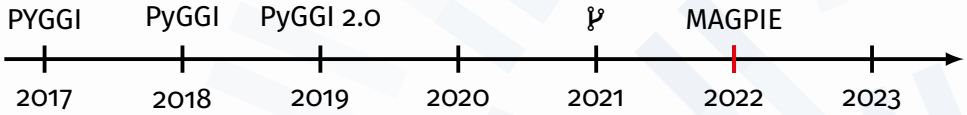
2021: fork (IEEE TEVC)

- local search (new)
- genetic programming
- validation algorithms

Blot and Petke, IEEE TEVC 2021

A Brief History

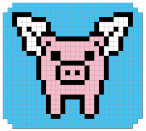


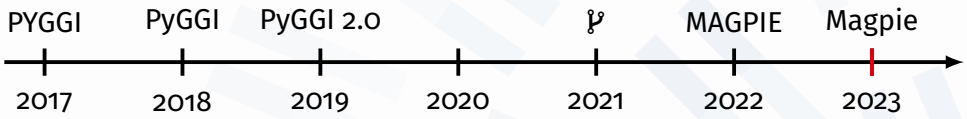


2022: MAGPIE (arXiv)

- parameter configuration (compiler/interpreter/software)
- mutation-testing mutations (numerical, operators)

A Brief History

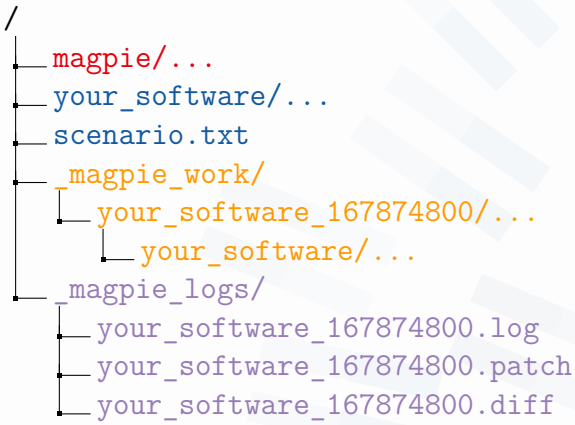




2023: Magpie

- almost complete rewrite
- syntactic “no-code” sugar

Magpie Structure



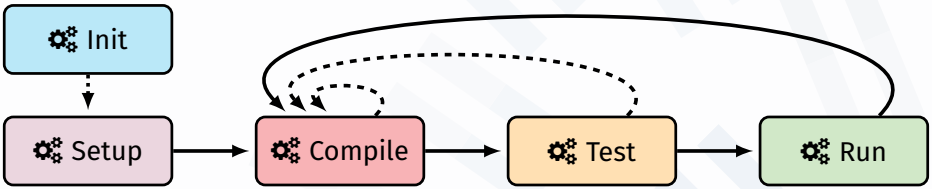
Not included for clarity: documentation, examples, tests, etc.

Scenario File

```
1  [software]
2  path = examples/triangle-py
3  target_files = triangle.py
4  fitness = time
5
6  init_cmd = bash init_slow.sh
7  test_cmd = pytest
8  run_cmd = python run_triangle.py
9  run_timeout = 1
10
11 [search]
12 max_steps = 100
13 max_time = 60
14 possible_edits =
15     LineReplacement LineInsertion LineDeletion
```

```
> python3 magpie local_search --scenario foo.txt
> python3 magpie minify_patch --scenario foo.txt --patch patch.txt
```

Software Multi-Step Evaluation



Configured in scenario file

- {step}_cmd
- {step}_timeout
- {step}_lengthout

Step-specific logging

- {step}_CLI_ERROR
- {step}_CODE_ERROR
- {step}_TIMEOUT
- {step}_LENGTHOUT
- {step}_PARSE_ERROR
- SUCCESS

Performance Indicators

Internally computed

- `time`
- `bloat_{lines,words,chars}`

Output-based (STDOUT, STDERR)

- `repair`: `/(\d+) (?:error|fail)/` (presets for JUnit, Pytest, Minitest)
- `posix_time`: `/real (\S+)/`
- `perf_instructions`: `/(\S+) instructions/`
- `perf_time`: `/(\S+) seconds time elapsed/`
- `output`: `/MAGPIE_OUTPUT: (\S+)/`

➔ *“easy to extend”*

Software Representation

LineModel

- LineDeletion
- LineInsertion
- LineReplacement
- LineMoving
- LineSwap

AstorModel

- StmtDeletion
- StmtInsertion
- StmtReplacement
- StmtMoving
- StmtSwap

XmlModel

- {...}Deletion
- {...}Insertion
- {...}Replacement
- {...}Setting
- {...}Moving
- {...}Swap

ConfigFileParamsModel

- ParamSetting

XML example

```
1  <cpp:include>#<cpp:directive>include</cpp:directive> <cpp:file>"ro
2
3  <comment type="line">// rotate three values</comment>
4  <function><type><name>void</name></type> <name>rotate</name><param
5
6    <comment type="line">// copy original values</comment>
7    <decl_stmt><decl><type><name>int</name></type> <name>tn1</name>
8
9    <comment type="line">// move</comment>
10   <expr_stmt><expr><name>n1</name> <operator>=</operator> <name>tn
11   <expr_stmt><expr><name>n2</name> <operator>=</operator> <name>tn
12   <expr_stmt><expr><name>n3</name> <operator>=</operator> <name>tn
13 </block_content>}</block></function>
```

Notes

- SrcML supports C, C++, C#, Java
- similar XML can be obtained using reflection and the Visitor pattern
- Magpie provides XML cleaning and processing utilities

https://www.srcml.org/

Algorithm Configuration Example

```
1  CLI_PREFIX = "-"
2  CLI_GLUE = "="
3  CLI_BOOLEAN = "prefix"
4
5  luby    {True, False}[True]   # Boolean
6  verb    {0, 1, 2}[1]          # categorical
7  phase-saving [0, 2][2]        # integer
8  var-decay     (0, 1)[0.95]    # float
9  gc-frac e(0, 65535)[0.2]      # exponential
10 rfirst  g[1, 65535][100]      # geometric
11 grow    g[-65535, 65535][0]
12
13 @sub-lim$flag      {True, False}[False] # @hidden parameter
14 sub-lim$unbounded {-1}[-1]              # $ignored suffix
15 sub-lim$bounded   g[0, 65535][1000]
16
17 sub-lim$unbounded | @sub-lim$flag == True # conditional
18 sub-lim$bounded   | @sub-lim$flag == False
```

Usage: `> ./minisat data/uf50-01.cnf -no-luby -verb=0 [...]`

Entry Points

```
> python3 magpie foo
```

Search algorithms

- `local_search`
- `genetic_programming`

Validation algorithms

- `revalidate_patch`
- `minify_patch`
- `ablation_analysis`

Helpers

- `show_patch`
- `show_locations`

```
> python3 foo
```

External scripts

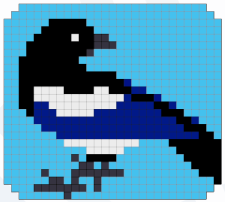
- `line_to_xml.py`
- `python_to_xml.py`
- `clear_xml.py`
- (more to be extracted from `core`)

Take Away

Automated Software Performance Improvement

Magpie

- tiny but full of functionalities!
- no knowledge of Python required!
- the best[*citation needed*] tool out there!



Try it now!

```
> git clone https://github.com/bloa/magpie.git
> cd magpie
> git checkout unstable
> python3 magpie local_search --scenario \
              examples/triangle-cpp/_magpie/scenario_slow.txt
```

This slide intentionally left blank

References I

Gabin An, Aymeric Blot, Justyna Petke, and Shin Yoo.

PyGGI 2.0: Language independent genetic improvement framework.

In *ESEC/FSE 2019*, pages 1100–1104, 2019.

Gabin An, Jinhan Kim, Seongmin Lee, and Shin Yoo.

PyGGI: Python General framework for Genetic Improvement.

In *KSC 2017*, pages 536–538, 2017.

Gabin An, Jinhan Kim, and Shin Yoo.

Comparing Line and AST granularity level for program repair using PyGGI.

In *GI@ICSE 2018*, pages 19–26, 2018.

References II

Aymeric Blot and Justyna Petke.

Empirical comparison of search heuristics for genetic improvement of software.

IEEE Trans. Evol. Comput., 25(5):1001–1011, 2021.

Aymeric Blot and Justyna Petke.

MAGPIE: Machine automated general performance improvement via evolution of software.

CoRR, abs/2208.02811, 2022.

William B. Langdon, Afnan Al-Subaihin, Aymeric Blot, and David Clark.

Genetic improvement of LLVM intermediate representation.

In *EuroGP 2023*, volume 13986 of *LNCS*, pages 144–159, 2023.

References III

William B. Langdon and Bradley J. Alexander.
Genetic improvement of OLC and H3 with magpie.
In *GI@ICSE 2023*, 2023.

William B. Langdon, Justyna Petke, Aymeric Blot, and David Clark.
Genetically improved software with fewer data caches misses.
In *GECCO 2023 companion*, 2023.

Justyna Petke, Saemundur O. Haraldsson, Mark Harman, William B. Langdon, David R. White, and John R. Woodward.
Genetic improvement of software: A comprehensive survey.
IEEE Trans. Evol. Comput., 22(3):415–432, 2018.