Automated Software Performance Improvement with Magpie

Aymeric Blot
Université de Rennes, France
April 27, 2025 – GI@ICSE 2025, Ottawa

Université de Rennes

Outline

Personal Background

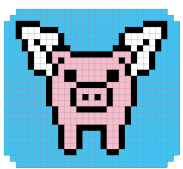
Aymeric Blot

- Lecturer @ Université de Rennes
- Research Fellow @ IRISA
- Member @ DiverSE (Inria/IRISA)
- Developing Magpie
- Contributed to PyGGI 2.0
- Developed MO-ParamILS

Université de Rennes

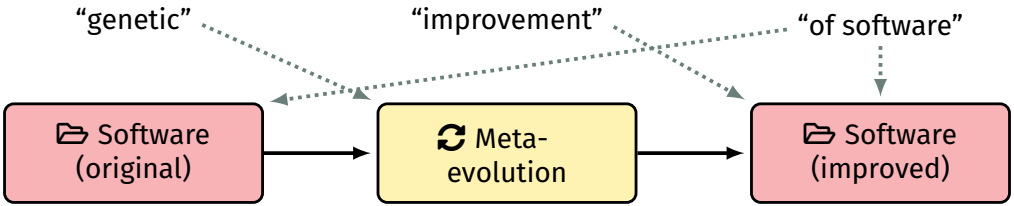
UMR IRISA Inria

DiverSE
Diversity-Centric Software Engineering



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

What is... Genetic Improvement of Software?

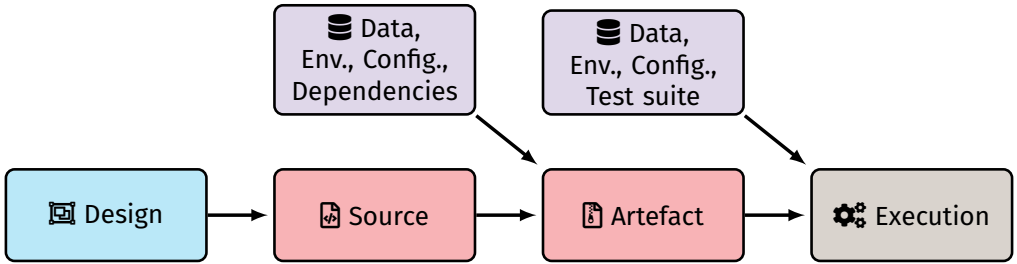


Core targets

- **Functional properties:** e.g., automated program repair
- **Non-functional properties:** performance improvement
  → e.g., execution time, energy/memory usage, solution quality...

What is... Software?

The description of an ultimately executable system?

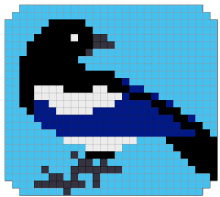


Observations

- Software development comprise many successive steps
- Most can be automatically conducted/optimised
- Most tools are dedicated to one specific use case only

What is... Magpie?

Machine Automated General Performance Improvement via Evolution (of softw...)

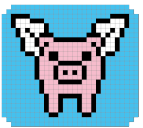


Magpie is...
a Python framework...
to model software...
and automate searching...
for improved software variants

Key points

- Free, libre, and open-source
- Small sized but very versatile
- Multi-purpose and trustworthy

A Brief History of Magpie



PYGGI — PyGGI — PyGGI 2.0 — MAGPIE — Magpie

2017 2018 2019 2020 2021 2022 2023 2024 2025

PYGGI An, Kim, and Yoo, KSC 2017

PyGGI An, Kim, Lee, and Yoo, GI@GECCO 2018

PyGGI 2.0 An, Blot, Petke, and Yoo, ESEC/FSE 2019

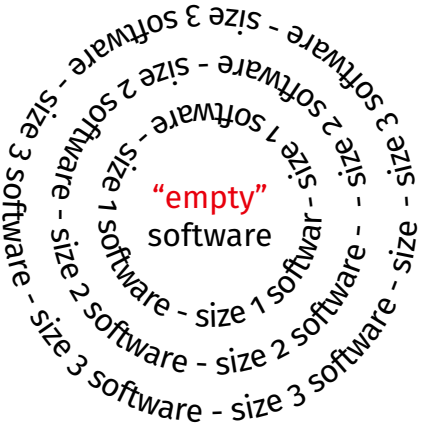
Blot and Petke, IEEE TEVC 2021

MAGPIE Blot and Petke, CoRR 2022

Outline

Software Search Space

Program Synthesis (from scratch): Nothing $\xrightarrow{\text{synthesis}}$ **New software**



Natural representation

Software (model)

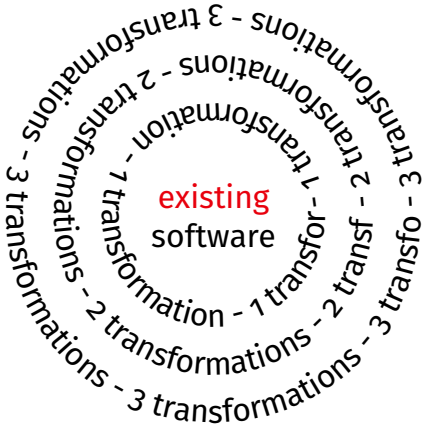
Key points

- Virtually infinite search space
- Hard to navigate
- Sparse: mostly irrelevant
+ Maximum expressiveness

Good software is far from the origin

Standing on the Shoulder of Giants

Genetic Improvement: Software $\xrightarrow{\text{evolution}}$ **New software**



Ad hoc representation

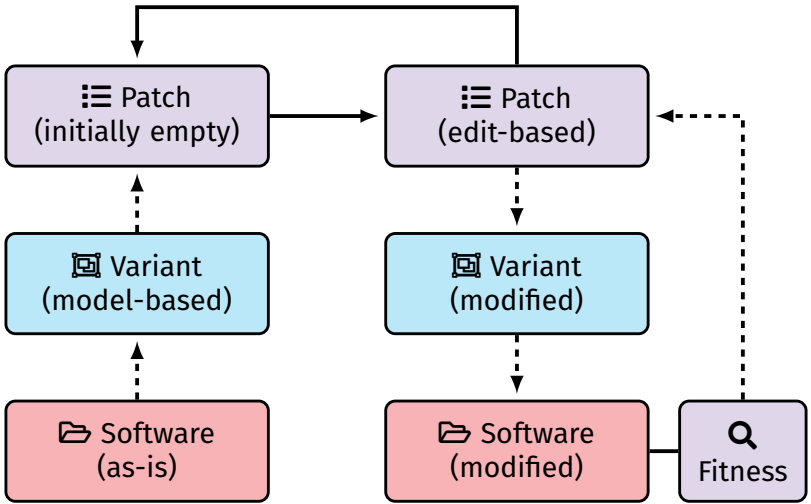
Sequence of transformations

Key points

- Still virtually infinite
- Still hard to navigate
- Reduced expressiveness
- \+ Centred on all interesting variants

Optimised software is “very close”

A Transformation-Centred Approach



Terminology in Magpie

Variant

A *collection* of **models** on which to apply **patches**

Model

An *internal representation* of a part of the target software

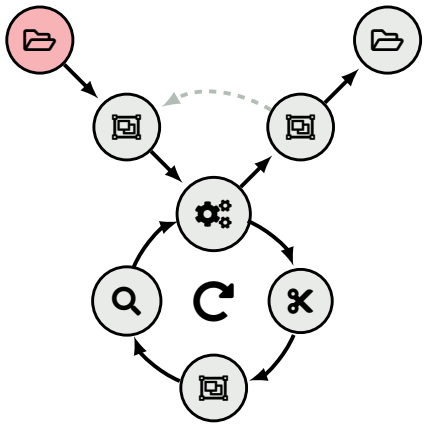
Patch

A *sequence* of **edits**

Edit

A single **model** *transformation*

Searching for Improved Variants



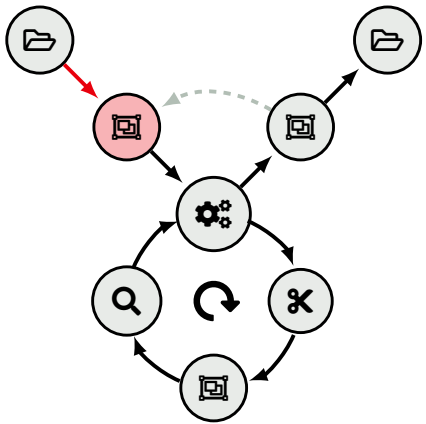
Standard search procedure

1. Start with existing software
2. Abstract parts on which to focus
3. Evolve a patch
   - 3.1 Add/remove edit
   - 3.2 Apply to create new variant
   - 3.3 Evaluate fitness function
4. Enjoy your improved software

Optional additional steps

- Test for generalisation
- Minimise patch size
- Manually assess semantics

Searching for Improved Variants



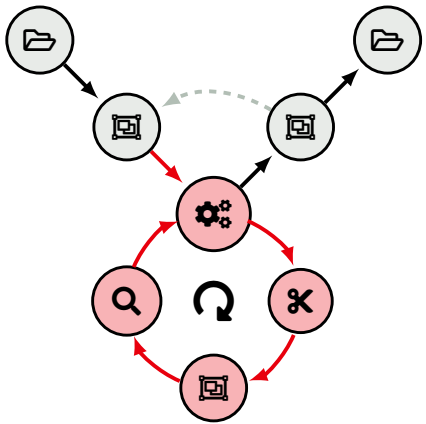
Standard search procedure

1. Start with existing software
2. Abstract parts on which to focus
3. Evolve a patch
   - 3.1 Add/remove edit
   - 3.2 Apply to create new variant
   - 3.3 Evaluate fitness function
4. Enjoy your improved software

Optional additional steps

- Test for generalisation
- Minimise patch size
- Manually assess semantics

Searching for Improved Variants



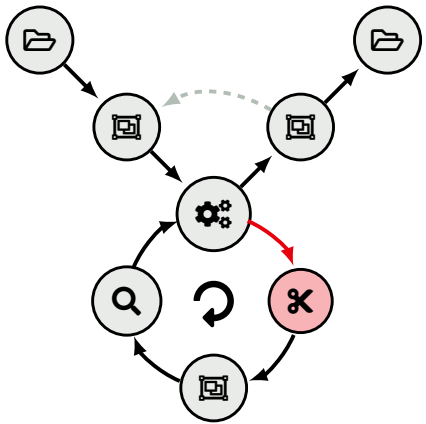
Standard search procedure

1. Start with existing software
2. Abstract parts on which to focus
3. Evolve a patch
   - 3.1 Add/remove edit
   - 3.2 Apply to create new variant
   - 3.3 Evaluate fitness function
4. Enjoy your improved software

Optional additional steps

- Test for generalisation
- Minimise patch size
- Manually assess semantics

Searching for Improved Variants



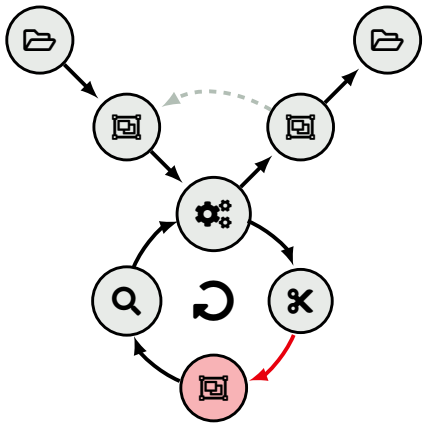
Standard search procedure

1. Start with existing software
2. Abstract parts on which to focus
3. Evolve a patch
   - 3.1 Add/remove edit
   - 3.2 Apply to create new variant
   - 3.3 Evaluate fitness function
4. Enjoy your improved software

Optional additional steps

- Test for generalisation
- Minimise patch size
- Manually assess semantics

Searching for Improved Variants



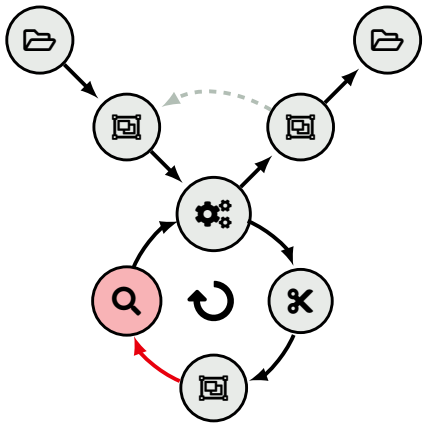
Standard search procedure

1. Start with existing software
2. Abstract parts on which to focus
3. Evolve a patch
   - 3.1 Add/remove edit
   - 3.2 Apply to create new variant
   - 3.3 Evaluate fitness function
4. Enjoy your improved software

Optional additional steps

- Test for generalisation
- Minimise patch size
- Manually assess semantics

Searching for Improved Variants



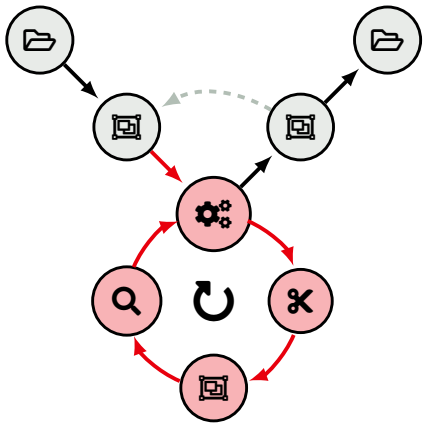
Standard search procedure

1. Start with existing software
2. Abstract parts on which to focus
3. Evolve a patch
   - 3.1 Add/remove edit
   - 3.2 Apply to create new variant
   - 3.3 Evaluate fitness function
4. Enjoy your improved software

Optional additional steps

- Test for generalisation
- Minimise patch size
- Manually assess semantics

Searching for Improved Variants



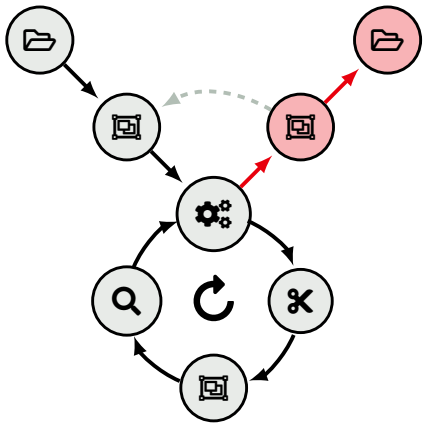
Standard search procedure

1. Start with existing software
2. Abstract parts on which to focus
3. Evolve a patch
   - 3.1 Add/remove edit
   - 3.2 Apply to create new variant
   - 3.3 Evaluate fitness function
4. Enjoy your improved software

Optional additional steps

- Test for generalisation
- Minimise patch size
- Manually assess semantics

Searching for Improved Variants



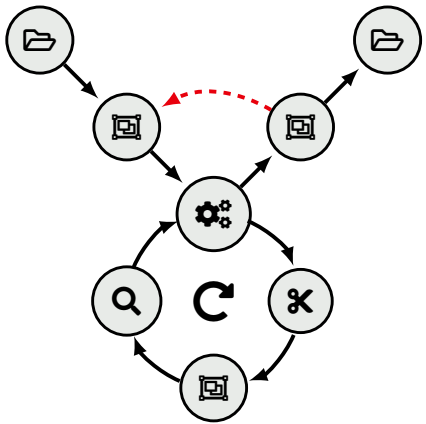
Standard search procedure

1. Start with existing software
2. Abstract parts on which to focus
3. Evolve a patch
   - 3.1 Add/remove edit
   - 3.2 Apply to create new variant
   - 3.3 Evaluate fitness function
4. Enjoy your improved software

Optional additional steps

- Test for generalisation
- Minimise patch size
- Manually assess semantics

Searching for Improved Variants



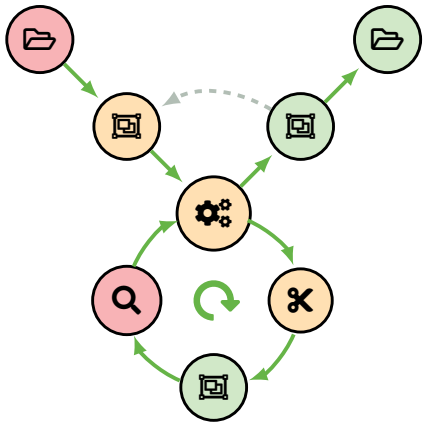
Standard search procedure

1. Start with existing software
2. Abstract parts on which to focus
3. Evolve a patch
   - 3.1 Add/remove edit
   - 3.2 Apply to create new variant
   - 3.3 Evaluate fitness function
4. Enjoy your improved software

Optional additional steps

- Test for generalisation
- Minimise patch size
- Manually assess semantics

Components' Origin



User-provided (software)

- Reference software
- Test suite
- Evaluation benchmark

User-specified (scenario)

- Models
- Search process
- Model transformations
- Performance indicator

Automated (Magpie)

- Procedure

Why Does it Work?

The "*competent programmer*" hypothesis (from MT)

Competent programmers write programs that are *almost* correct

The "*plastic surgery*" hypothesis (from APR)

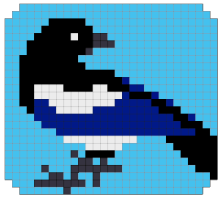
1. Program source code changes occurring during development contain snippets that already exist in the codebase
2. These snippets can be efficiently found and exploited

The "*your software is not unique*" observation (from *me*)

There are *infinitely many* equivalent software, in a *fractal* fashion

Outline

Magpie in a Nutshell



Magpie

- Modular Python framework for GI/MT/AC/CO
- Hack-friendly, for researchers!
- User-friendly, for developers!

Key points

- Free, libre, and open-source → MIT license, on GitHub
- Small sized but very versatile → ≈ 4500 loc, *many functionalities*
- Multi-purpose and trustworthy → F/NF properties, scenario-based

Magpie's Architecture

```
magpie/ (← the Git repository)
├── docs/ (← Diátaxis-based - 🔗 https://diataxis.fr/)
├── examples/
│   └── triangle-c/ (← target software)
│       └── _magpie/ (← scenario files)
├── magpie/
│   ├── core/ (← base classes + scenario config ≈ 1600 loc)
│   ├── models/ (← line, xml, config, astor ≈ 1300 loc)
│   ├── algos/ (← LS, GP, validation, ablation ≈ 800 loc)
│   ├── fitness/ (← time, repair, bloat, output ≈ 200 loc)
│   ├── bin/ (← entry points ≈ 200 loc)
│   └── __main__.py (← main entry point)
└── tests/
```

Suggested Project Structure

```
/
├── magpie/ (← the one containing "__main__.py")
├── your_software/
├── scenario.txt
├── _magpie_work/ (← automatically generated)
│   └── your_software_167874800/
└── _magpie_logs/ (← automatically generated)
    ├── your_software_167874800.log
    ├── your_software_167874800.patch
    └── your_software_167874800.diff
```

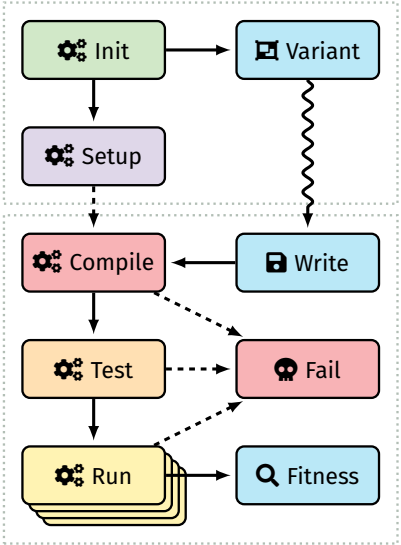
In practice

```
> python3 magpie local_search --scenario scenario.txt
> python3 magpie minify_patch --scenario s.txt --patch p.txt
```

Scenario File: INI Configuration File

```
1  [software]
2  path = examples/triangle-rb
3  target_files =
4      triangle.rb
5  fitness = repair
6
7  init_cmd = bash init_bug.sh
8  test_cmd = ruby test_triangle.rb
9
10 [search]
11 target_fitness = 0
12 max_steps = 100
13 possible_edits =
14     LineReplacement
15     LineInsertion
16     LineDeletion
```

Evaluation Pipeline



At the start of the search

- **init:** environment preparation
  → **always executed**
- **setup:** pre-processing
  → **only when required**

For every variant

- **compile:** test/run preparation
- **test:** semantic check
- **run:** fitness computation
  → **can be repeated**

Note

All five steps can be left empty!

Magpie's Models

List-based

- **LineModel** → line replacement, insertion, deletion

Tree-based

- **XmlModel** → node replacement, insertion, deletion, node text setting, wrapping
- **SrcmlModel** → (same)
- **AstorModel** → statement replacement, insertion, deletion

Dictionary-based

- **ParamFileConfigModel** → parameter setting

Note

Model association is done by default through file extension

Total size: approximately 1300 loc (models + transformations)

Magpie's Algorithms

LS-based
- FirstImprovement
- BestImprovement
- TabuSearch
- WorstImprovement

GP-based
- GeneticProgramming1Point
- GeneticProgramming2Point
- GeneticProgrammingConcat
- GeneticProgrammingUniformConcat
- GeneticProgrammingUniformInter

Validation-based
- ValidMinify
- ValidTest
- ValidSingle = DebugSearch

Other
- AblationAnalysis
- RandomSearch
- RandomWalk

Total size: approximately 800 loc

Magpie's Fitness Functions

Execution time

- `time`
- `posix_time` `/real (\S+)/` ∈ run.stderr
- `perf_time` `/(\S+) seconds time elapsed/` ∈ run.stderr
- `perf_instructions` `/(\S+) instructions/` ∈ run.stderr
- new: `gnu_time<key>`, `perf<key>` key line ∈ run.stderr

Program repair

- `repair` currently supports JUnit, pytest, and minitest
  e.g., `/Failures: (\d+)/` and `/Tests run: (\d+)/` ∈ test.stdout

Misc.

- `bloat_{lines,words,chars}`
- `output` `/MAGPIE_FITNESS: (\S+)/` ∈ run.stdout

Total size: approximately 200 loc

Magpie's Entry Points

Usage

```
> python3 magpie [[magpie/]{bin,scripts}/]ENTRY[.py]
> mv magpie/{bin,scripts}/ENTRY.py .; python3 ENTRY.py
> python3 -m magpie.{bin,scripts}.ENTRY
```

Search

- bin/local_search.py
- bin/genetic_programming.py

Validation

- bin/show_patch.py
- bin/revalidate_patch.py
- bin/minify_patch.py
- bin/ablation_analysis.py

Misc.

- bin/show_locations.py

Scripts

- scripts/clear_xml.py
- scripts/line_to_xml.py
- scripts/python_to_xml.py

Magpie 1.2 in a Nutshell

Added

- support for multi-objective optimisation, maximisation
- templated fitness functions (e.g., `perf<instructions>`)
- templated edits (e.g., `XmlNodeReplacement<stmt>`)

Changed

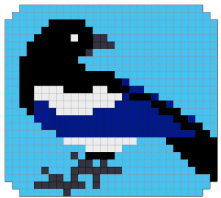
- fitness functions are now first-class citizens
- edit classes renamed for consistency
- patch ingredients changed for consistency

Fixed

- `repair` support for jUnit
- GP failing on very small search spaces
- instance batch output processing

Outline

Magpie in Practice



Magpie

- Modular Python framework for GI/MT/AC/CO
- Hack-friendly, for researchers!
- User-friendly, for developers!

Try it now!

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

First Steps: triangle-c

```
magpie/ (← the Git repository)
└── examples/
    └── triangle-c
        ├── triangle.c
        ├── triangle.h
        ├── run_triangle.c
        ├── test_triangle.c
        ├── makefile
        ├── init_slow.sh
        └── _magpie/
            ├── scenario_slow.txt
            ├── triangle_slow.c.diff
            ├── triangle_slow.h.diff
            └── triangle_slow.c.xml
```

The Scenario

examples/triangle-c/_magpie/scenario_slow.txt

```
[software]
path = examples/triangle-c
target_files = triangle.c.xml
fitness = time

init_cmd = bash init_slow.sh
compile_cmd = make test_triangle run_triangle
test_cmd = ./test_triangle
run_cmd = ./run_triangle
run_timeout = 1

[search]
max_steps = 100
max_time = 60
possible_edits =
    XmlNodeDeletion<stmt>
    XmlNodeReplacement<stmt>
    XmlNodeInsertion<stmt,block>
```

The Preparation

examples/triangle-c/init_slow.sh

```
1 #!/usr/bin/env bash
2
3 patch triangle.c _magpie/triangle_slow.c.diff
4 patch triangle.h _magpie/triangle_slow.h.diff
5 #srcml triangle.c > _magpie/triangle_slow.c.xml
6 mv _magpie/triangle_slow.c.xml triangle.c.xml
```

`init_cmd` **is executed once, every time**

examples/triangle-cpp/setup.sh

```
1 #!/usr/bin/env bash
2
3 rm -rf build
4 mkdir build
5 cd build
6 cmake ..
```

`setup_cmd` **is executed once, only "when necessary"**

The Bug

examples/triangle-c/_magpie/triangle_slow.c.diff

```
--- triangle.c
+++ triangle_slow.c
@@ -1,9 +1,16 @@
 #include "triangle.h"

+void delay() {
+  const struct timespec ms = {0, 0.001*1e9}; //(0.001s)
+  nanosleep(&ms,NULL); /*ignores possible errors*/
+}
+
 int classify_triangle(double a, double b, double c) {
   double tmp;

+  delay();
+
   // sort the sides so that a <= b <= c
   if(a > b) {
```

The XML AST Representation

examples/triangle-c/_magpie/triangle.c (excerpt)

```
1   // sort the sides so that a <= b <= c
2   if(a > b) {
3     tmp = a;
4     a = b;
5     b = tmp;
6   }
```

examples/triangle-c/_magpie/triangle_slow.c.xml (excerpt)

```
1   <comment type="line">// sort the sides so that a &lt;= b &l
2   <if_stmt><if>if<condition>(<expr><name>a</name> <operator>&
3     <expr_stmt><expr><name>tmp</name> <operator>=</operator>
4     <expr_stmt><expr><name>a</name> <operator>=</operator> <n
5     <expr_stmt><expr><name>b</name> <operator>=</operator> <n
6   </block_content>}</block></if></if_stmt>
```

https://www.srcml.org/ **(supports C, C++, C#, Java)**

XML Processing

Default rules

```
1 [srcml]
2 rename =
3     stmt: break continue decl_stmt do expr_stmt for goto if r
4     number: literal_number
5 focus = block stmt operator_comp operator_arith number
6 internodes = block
7 process_pseudo_blocks = True
8 process_literals = True
9 process_operators = True
```

- `rename`: dictionary of renaming rules
- `focus`: list of tags to keep
- `internodes`: list of tags to process for future note insertion
- `process_...`: bespoke SrcML rules

Running Magpie!

```
› python3 magpie local_search \
      --scenario examples/triangle-c/_magpie/scenario_slow.txt
```

```
1  ==== SEARCH: FirstImprovement ====
2  ~~~~ WARMUP ~~~~
3  WARM   SUCCESS            0.07 (--) [0 edit(s)]
4  WARM   SUCCESS            0.08 (--) [0 edit(s)]
5  WARM   SUCCESS            0.08 (--) [0 edit(s)]
6  REF    SUCCESS            0.08 (--) [0 edit(s)]
7
8  ~~~~ START ~~~~
9  1      TEST_CODE_ERROR    None (--) [1 edit(s)]
10 2      SUCCESS           *0.08 (96.18%) [1 edit(s)]
11 3      SUCCESS           *0.07 (92.74%) [2 edit(s)]
12 4      SUCCESS            0.08 (102.29%) [1 edit(s)]
13 5      SUCCESS            0.08 (96.18%) [1 edit(s)] [cached]
14 6      TEST_CODE_ERROR    None (--) [3 edit(s)]
15 7      TEST_CODE_ERROR    None (--) [3 edit(s)]
16 8      SUCCESS            0.15 (188.92%) [3 edit(s)]
```

Multi-Step Evaluation and Logging

Configured in scenario file

- `{step}_cmd`
- `{step}_timeout`
- `{step}_lengthout`
- `batch_timeout`
- `batch_lengthout`

Step-specific logging

- `{step}_CLI_ERROR`
- `{step}_CODE_ERROR`
- `{step}_TIMEOUT`
- `{step}_LENGTHOUT`
- `{step}_PARSE_ERROR`
- `SUCCESS` (i.e., fitness value)

Note: Fails during warmup trigger automated self-diagnostic

Finding Variants

```
24      TEST_CODE_ERROR        None (--) [4 edit(s)]
25      COMPILE_CODE_ERROR     None (--) [4 edit(s)]
26      SUCCESS               *0.00 (4.08%) [2 edit(s)]
^C~~~~ END ~~~~

==== REPORT ====
Termination: keyboard interrupt
Log file: /home/aymeric/git/magpie/_magpie_logs/triangle-c_17
Patch file: _magpie_logs/triangle-c_1712601481.patch
Diff file: _magpie_logs/triangle-c_1712601481.diff
Reference fitness: 0.08
Best fitness: 0.00

==== BEST PATCH ====
XmlNodeInsertion<stmt,block>(('triangle.c.xml', '_inter_block

==== DIFF ====
--- before: triangle.c
+++ after: triangle.c
```

Final Diff

```
--- before: triangle.c
+++ after: triangle.c
@@ -2,7 +2,7 @@

 void delay() {
    const struct timespec ms = {0, 0.001*1e9}; //tv_sec=0, tv_
-   nanosleep(&ms,NULL); /*ignores possible errors*/
+    /*ignores possible errors*/
 }

 int classify_triangle(double a, double b, double c) {
@@ -40,6 +40,7 @@
    if(a == b || b == c)/*auto*/{

      return ISOSCELES;
+     return EQUILATERAL;
    }/*auto*/
    return SCALENE;
 }
```

What to do with a Patch?

Look at it (and possibly `--keep` it)

```
› python3 magpie show_patch \
      --scenario examples/triangle-c/_magpie/scenario_slow.txt \
      --patch _magpie_logs/triangle-c_1712601481.patch \
      --keep
```

Revalidate it (possibly on a different scenario)

```
› python3 magpie revalidate_patch --scenario ...  --patch ...
```

Minify it (possibly on a different scenario)

```
› python3 magpie minify_patch --scenario ...  --patch ...
```

Study it

```
› python3 magpie ablation_analysis --scenario ...  --patch ...
```

Changing the Fitness Function

Execution time (Python) → noisy, precise, "free"

- [software] `fitness = time`
- [software] `run_cmd = ./run_triangle`

Execution time (POSIX) → noisy, not very precise, UNIX

- [software] `fitness = posix_time`
- [software] `run_cmd = /usr/bin/time -p ./run_triangle`

Execution time (perf) → noisy, very precise, linux

- [software] `fitness = perf<elapsed>`
- [software] `run_cmd = perf stat ./run_triangle`

CPU instructions (perf) → "deterministic", extremely precise, linux/HW

- [software] `fitness = perf<instructions>`
- [software] `run_cmd = perf stat ./run_triangle`

Changing More Scenario Specifics

Setting a random seed

- `[magpie] seed = ...` (integer)

Configuring warmup

- `[search] warmup = ...` (strictly positive integer)
- `[search] warmup_strategy = ...` ({`last`, `min`, `max`, `mean`, `median`})

Configuring the stopping condition

- `[search] max_steps = ...` (integer or empty)
- `[search] max_time = ...` (integer or empty)
- `[search] target_fitness = ...` (integer or empty)

Getting Serious: MiniSAT

```
magpie/ (← Git)
└── examples/
    └── minisat
        ├── init.sh
        ├── compile.sh
        ├── test.sh
        ├── run.sh
        ├── minisat.params (← runtime configuration space)
        ├── data/ (← input instances)
        └── _magpie/
            ├── scenario.txt
            └── Solver.cc.xml
```

Preliminary Setup (1/3)

examples/minisat/init.sh

```
#!/usr/bin/env bash

CACHE=$MAGPIE_ROOT/_magpie_cache
ARCHIVE=minisat-2.2.0.tar.gz

# download and cache archive
mkdir -p $CACHE
if ! test -f $CACHE/$ARCHIVE; then
    wget "http://minisat.se/downloads/$ARCHIVE"
    mv $ARCHIVE $CACHE
fi

# extract, patch, and setup pre-computed XML AST
tar xzf $CACHE/$ARCHIVE --strip-components=1
patch core/Dimacs.h _magpie/dimacs.diff
# srcml core/Solver.cc > _magpie/Solver.cc.xml
cp _magpie/Solver.cc.xml core
```

Preliminary Setup (2/3)

examples/minisat/test.sh (excerpt)

```
1  #!/bin/sh
2  ARGV=$@
3
4  my_test() {
5      FILENAME=$1 ; EXPECTED=$2
6      ./simp/minisat $FILENAME $ARGV > /dev/null
7      RETURN=$?
8      if [ $RETURN -ne $((EXPECTED)) ]; then
9          echo "FAILED␣ON␣FILE:" $FILENAME
10         echo "GOT:" $RETURN
11         echo "EXPECTED:" $EXPECTED
12         exit -1
13     fi
14 }
15
16 my_test data/uf50-01.cnf 10
17 my_test data/uf50-02.cnf 10
18 my_test data/uuf50-01.cnf 20
19 my_test data/uuf50-02.cnf 20
```

Preliminary Setup (3/3)

examples/minisat/run_fixed.sh (excerpt)

```
1  #!/bin/sh
2
3  ./simp/minisat data/uf150-01.cnf $@
4  ./simp/minisat data/uf150-02.cnf $@
5  ./simp/minisat data/uf200-01.cnf $@
6  ./simp/minisat data/uf200-02.cnf $@
7  ./simp/minisat data/uf250-01.cnf $@
8  ./simp/minisat data/uf250-02.cnf $@
9
10 ./simp/minisat data/uuf150-01.cnf $@
11 ./simp/minisat data/uuf150-02.cnf $@
12 ./simp/minisat data/uuf200-01.cnf $@
13 ./simp/minisat data/uuf200-02.cnf $@
14 ./simp/minisat data/uuf250-01.cnf $@
15 ./simp/minisat data/uuf250-02.cnf $@
16
17 exit 0
```

Use Case 1: Line-level Evolution

examples/minisat/_magpie/scenario_runtime_line.txt

```
[software]
path = examples/minisat
target_files = core/*.cc
fitness = time

init_cmd = bash init.sh
setup_cmd = bash compile.sh
compile_cmd = bash compile.sh
test_cmd = bash test.sh
run_cmd = bash run_fixed.sh

[search]
max_steps = 100
possible_edits =
    LineReplacement
    LineInsertion
    LineDeletion
```

Use Case 2: AST-level Evolution

examples/minisat/_magpie/scenario_runtime_xml1.txt

```
1  [software]
2  path = examples/minisat
3  target_files = core/Solver.cc.xml
4  fitness = time
5
6  init_cmd = bash init.sh
7  setup_cmd = bash compile.sh
8  compile_cmd = bash compile.sh
9  test_cmd = bash test.sh
10 run_cmd = bash run_fixed.sh
11
12 [search]
13 max_steps = 100
14 possible_edits =
15     XmlNodeDeletion<stmt>
16     XmlNodeReplacement<stmt>
17     XmlNodeInsertion<stmt,block>
```

Use Case 3: Parameter Tuning

examples/minisat/_magpie/scenario_runtime_config1.txt

```
1  [software]
2  path = examples/minisat
3  target_files = minisat_simplified.params
4  fitness = time
5
6  init_cmd = bash init.sh
7  setup_cmd = bash compile.sh
8  compile_cmd =
9  test_cmd = bash test.sh
10 run_cmd = bash run_fixed.sh
11
12 [search]
13 max_time = 60
14 possible_edits = ParamSetting
```

Parameter Configuration File

examples/minisat/minisat_simplified.params (excerpt)

```
1  CLI_PREFIX = "-"
2  CLI_GLUE = "="
3  CLI_BOOLEAN = "prefix"
4
5  # core
6  luby       {True, False}[True]
7  rnd-init   {True, False}[False]
8  gc-frac    e(0, 65535)[0.2]
9  rinc       e(1, 65535)[2]
10 var-decay  (0, 1)[0.95]
11
12 phase-saving [0, 2][2]
13 ccmin-mode    [0, 2][2]
14 rfirst        g[1, 65535][100]
```

Examples: "`-luby`", "`-no-luby`", "`-gc-frac=0.2`", **etc**

More Syntax

```
# magic constants
CLI_...
TIMING="setup compile"

# categorical parameters
name {value1, value2, value3}[value1]
name {True, False, None}[None]

# numerical parameters
name (0, 1.0)[0] # uniform continuous
name e(0, 1.0)[0] # exponential
name [0, 100][0] # uniform discrete
name g[-999999999, 999999999][0] # geometric

# and more!
name1 | name2 == True # conditional
{name1 == True, name2 == True} # forbidden combination
@name {True, False}[True] # invisible parameter
name$suffix {True, False}[True] # invisible suffix
```

Use Case 4: Evolving at Multiple Granularities

```
1  [software]
2  target_files =
3      gcc.params
4      minisat.params
5      **/*.cc.xml
6
7  [search]
8  possible_edits =
9      ParamSetting
10     SrcmlNumericSetting
11     XmlNodeDeletion<stmt>
12     XmlNodeReplacement<stmt>
13     XmlNodeInsertion<stmt,block>
```

How are edits generated?

1. Select an edit type — from `possible_edits`
2. Identify target models and locations — potentially multiple times
3. Instantiate the edit — with additional data as needed

Use Case 5: Batch Processing

examples/minisat/_magpie/scenario_runtime_batch1.txt (excerpt)

```
1 [software]
2 run_cmd = bash run_single.sh {INST}
3
4 [search]
5 batch_instances =
6     file:data/inst_sat.txt
7     ___
8     file:data/inst_unsat.txt
9 batch_sample_size = 4
```

examples/minisat/run_single.sh

```
1 #!/bin/sh
2
3 ./simp/minisat $@
4 exit 0
```

Outline

The Magpie Wish List

From a tooling perspective

- Interactive search visualisation
- Interactive model visualisation
- Fine-grained user interaction

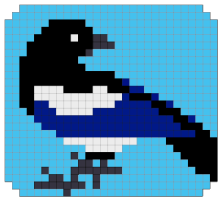
→ **Make the process more transparent and engaging**

From a research perspective

- Fault localisation
- Semantic search guidance
- Performance prediction
- Two-way explanations

→ **Increase efficiency, trust, and usability**

Final Words



Magpie, One Framework to Rule Them All!

- Modular Python framework for GI/MT/AC/CO
- Hack-friendly, for researchers!
- User-friendly, for developers!

Use it now!

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

Please do not hesitate to reach out for support!