Magpie

Machine Automated General Performance Improvement via Evolution of Software

Aymeric Blot

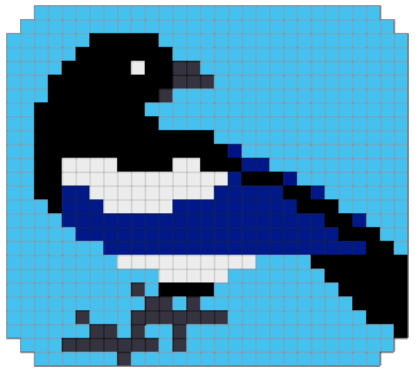
Université du Littoral Côte d'Opale

March 27, 2023 — 63rd CREST Open Workshop

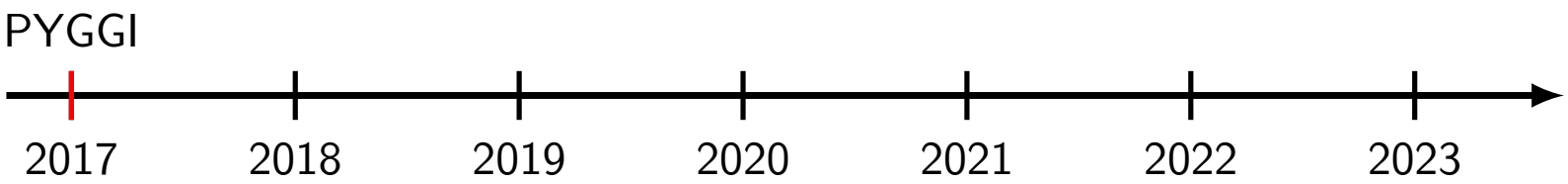
Magpie

In a nutshell

- Python framework
- Purposely very small (∼3kloc)
- Enable GI *and more*
- Free on GitHub
- *User-friendly* and *research-friendly*



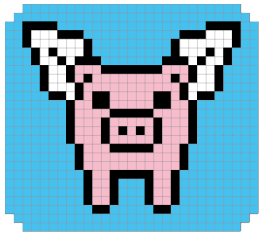
A Brief History

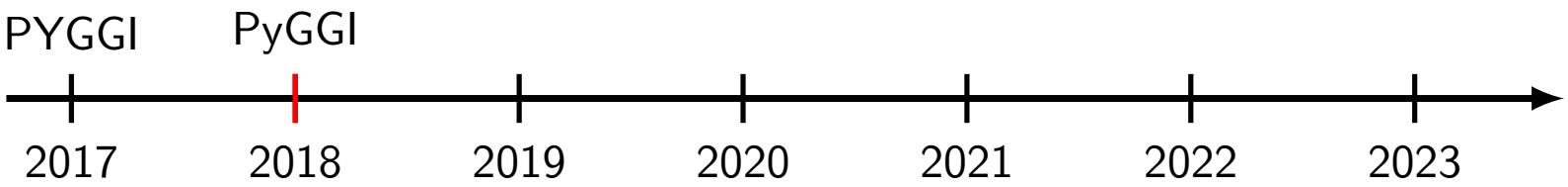


2017: PYGGI (KSC)

- Lines of code (deletion, insertion, replacement)
- Local search
- Program repair, execution time

A Brief History

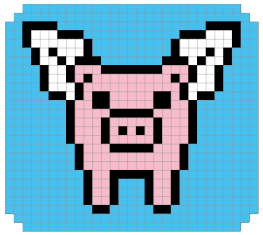


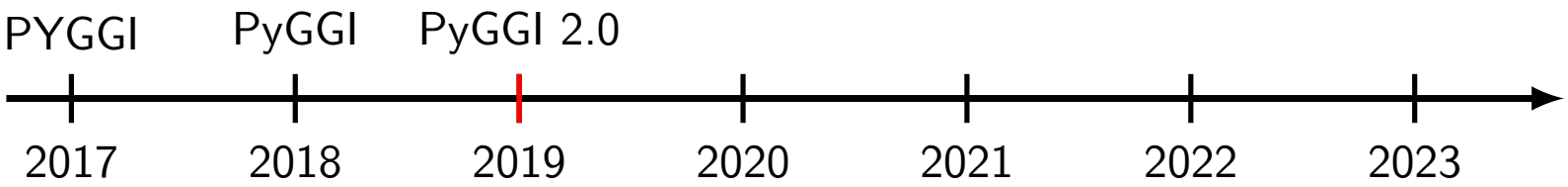


2018: PyGGI (GI@ICSE)

- Python statement AST (deletion, insertion, replacement)

A Brief History

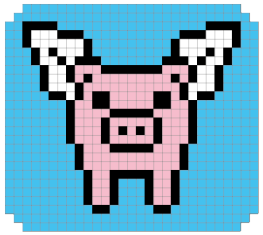


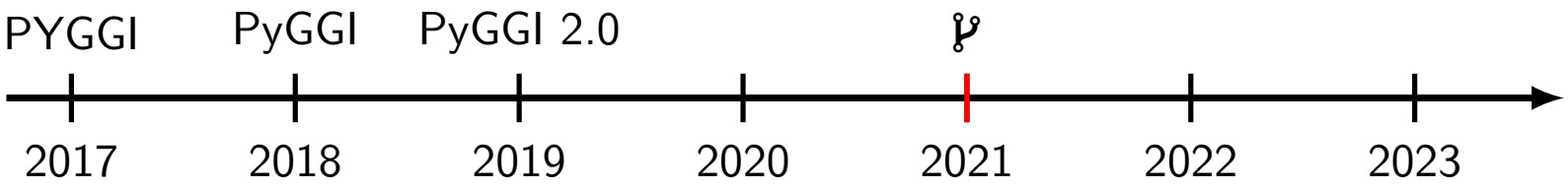


2019: PyGGI 2.0 (ESEC/FSE)

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

A Brief History

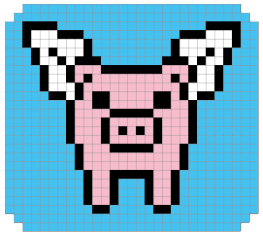


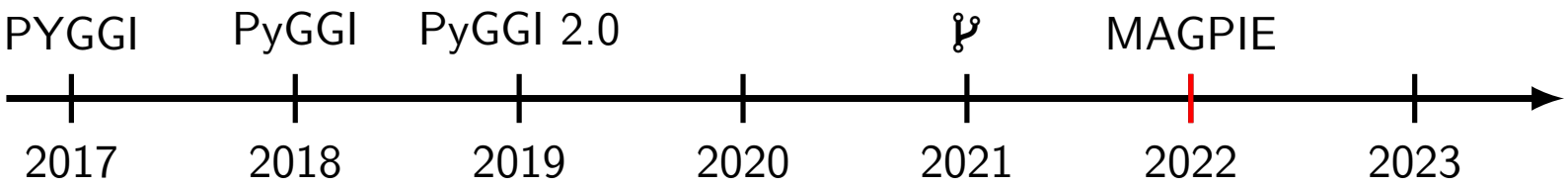


2021: fork (IEEE TEVC)

- Local search (new)
- Genetic Programming
- Validation algorithms

A Brief History

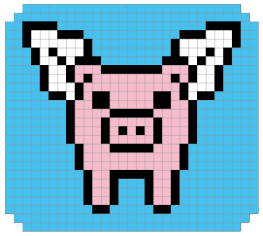


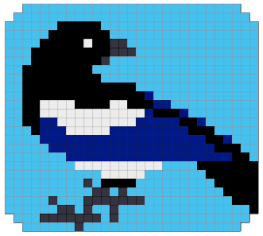


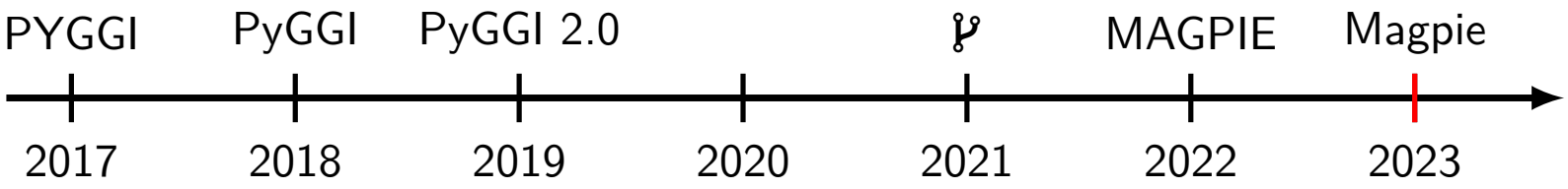
2022: MAGPIE (arXiv)

- Parameter configuration (compiler/interpreter/software)
- Mutation-testing mutations (numerical, operators)

A Brief History



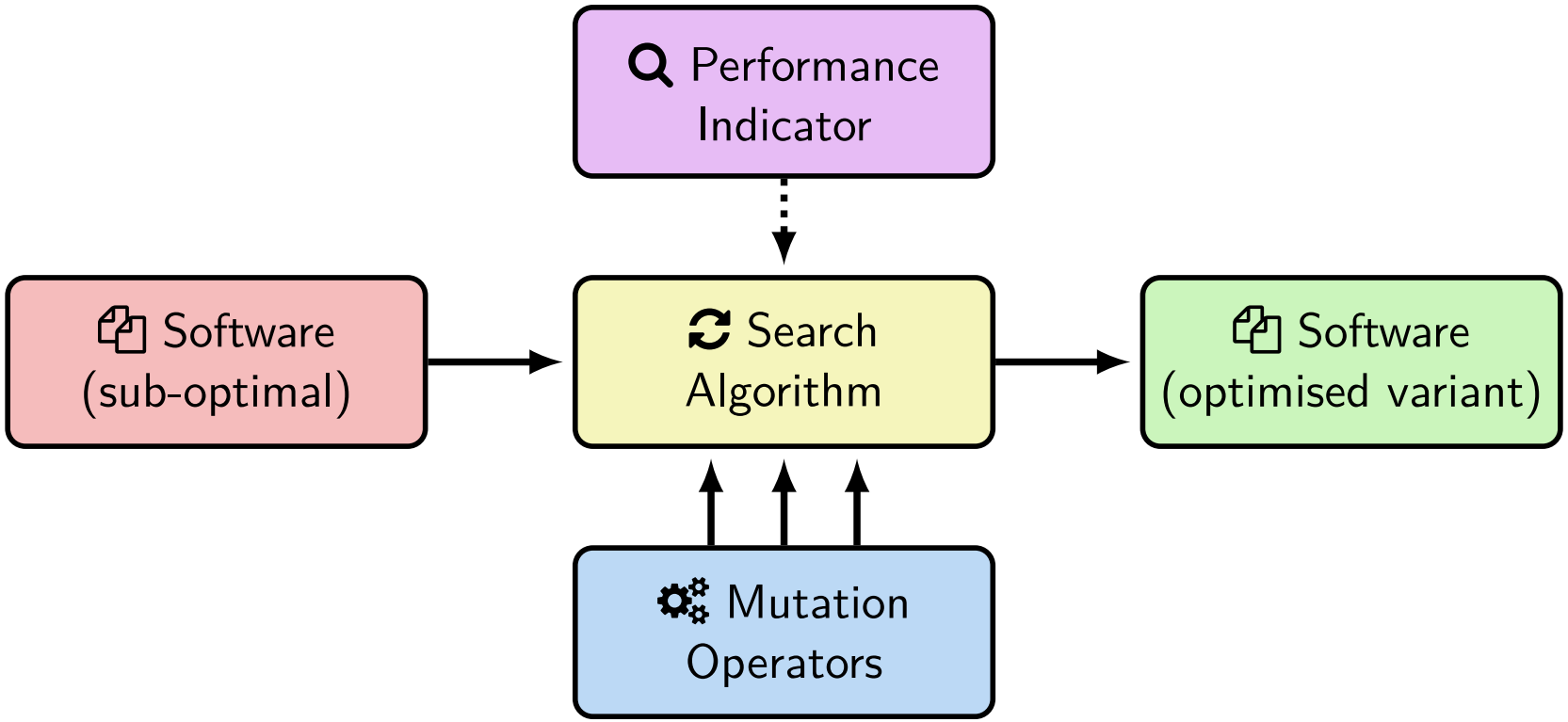




2023: Magpie

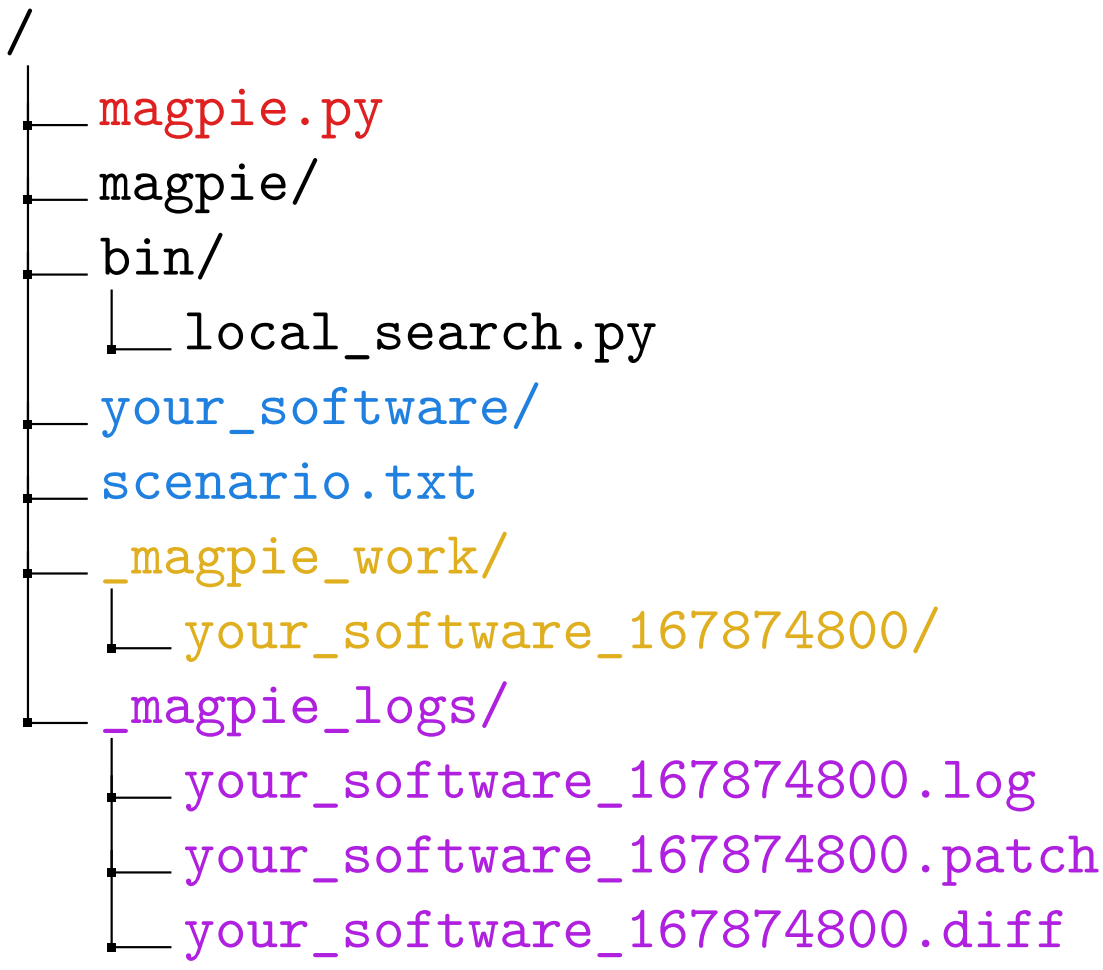
- Almost complete rewrite
- Syntactic “no-code” sugar

Evolving Software using Magpie




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

Magpie Structure

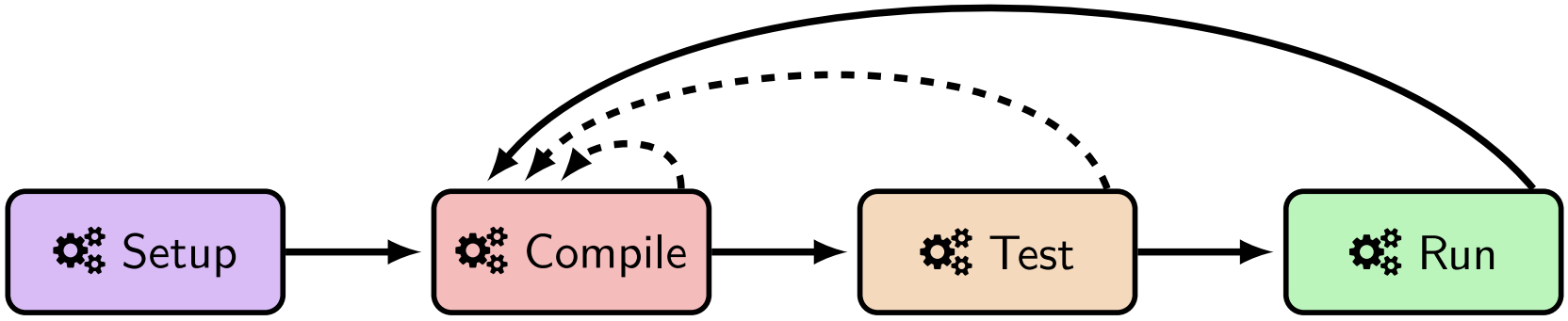


Not included for clarity: documentation, tests, etc.

Scenario File

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

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 Indicator

Out-of-the-box:

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

Output-specific:

- **`repair`**: `/(\d+) (?:error|fail)/` $\in$ test.stdout
- **`posix_time`**: `/real (\S+)/` $\in$ run.stderr
- **`perf_instructions`**: `/(\S+) instructions/` $\in$ run.stderr
- **`perf_time`**: `/(\S+) seconds time elapsed/` $\in$ run.stderr
- **`output`**: `/MAGPIE_OUTPUT: (\S+)/` $\in$ run.stdout
- *"easy to extend"*

Software Representation

LineEngine

- `LineDeletion`
- `LineInsertion`
- `LineReplacement`
- `LineMoving`

SrcmlEngine

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

ConfigFileParamsEngine

- `ParamSetting`

AstorEngine

- `StmtDeletion`
- `StmtInsertion`
- `StmtReplacement`
- `StmtMoving`

Advanced Scenario File

```
1  [software]
2  target_files = core/Solver.cc.xml
3                 minisat.params
4                 gcc.params
5  engine_config = *.xml : [srcml]
6                  minisat.params : [params.minisat]
7                  gcc.params : [params.gcc]
8
9  [srcml]
10 rename = stmt: break continue decl_stmt do expr_stmt for go
11          number: literal_number
12 focus = block stmt number operator_comp operator_arith
13 internodes = block
14 process_pseudo_blocks = True
15 process_literals = True
16 process_operators = True
17
18 [params.minisat]
19 timing = test run
20 [params.gcc]
21 timing = compile
```

Custom Code

```
[magpie]
import = custom.py

[search]
possible_edits =
    TypeReplacement
    TypeSetting
```

```
class TypeReplacement(NodeReplacement):
    NODE_TYPE = 'type'

class TypeSetting(TextSetting):
    NODE_TYPE = 'type'
    CHOICES = ['float', 'double', 'int']

magpie.xml.edits.append(TypeReplacement)
magpie.xml.exits.append(TypeSetting)
```

Parameter Configuration

```
 1 CLI_PREFIX = "-"
 2 CLI_GLUE = "="
 3 CLI_BOOLEAN = "prefix"
 4
 5 luby    {True, False}[True]   # categorical
 6 verb    {0, 1, 2}[1]
 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$flag      {True, False}[False] # @hidden
14 sub-lim$unbounded {-1}[-1]               # $ignored
15 sub-lim$bounded   g[0, 65535][1000]
16
17 sub-lim$unbounded | @sub-lim$flag == True # conditional
18 sub-lim$bounded   | @sub-lim$flag == False
```

Entry Points

Search

- `local_search.py`
- `genetic_programming.py`

Validation

- `show_patch.py`
- `revalidate_patch.py`
- `minify_patch.py`
- `ablation_analysis.py`

Misc.

- `show_locations.py`
- `line_xml.py`

What Now?

More content

- Batch-size support
- Multi-objective support
- New representations/transformations

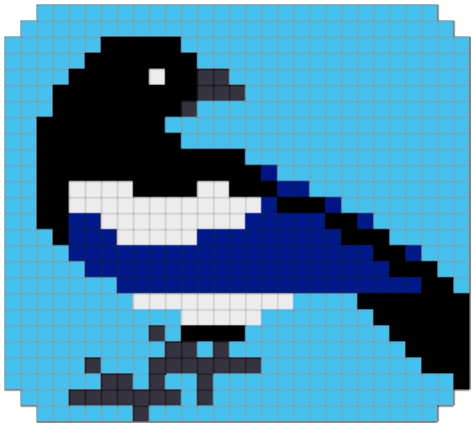
Documentation

- Tutorials
- More tests
- More comments

Take-away Message

Magpie

- Tiny but full of functionalities!
- No knowledge of Python required!
- `github.com/bloa/magpie`



Try it now!

```
> git clone https://github.com/bloa/magpie.git
> python3 magpie.py local_search --scenario
      examples/scenario/triangle-cpp_runtime.txt
```

Selected References

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.

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.