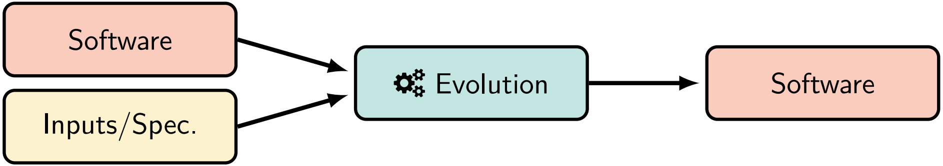
Evaluation of Genetic Improvement Tools for Improvement of Non-functional Properties of Software

Shengjie Zuo **Aymeric Blot** Justyna Petke

University College London

GI@GECCO'22 — 09 July 2022

Genetic Improvement (GI)



Challenges: Automated refactorisation, performance improvement
Motivation: Hidden flaws, specification changes, code rot, ...

Functional properties (FP)

- Automated bug fixing
- Code transplantation

Non-functional properties (NFP)

- Execution time
- Memory/energy usage
- Output quality
- Code size, attack surface

http://geneticimprovementofsoftware.com/

Motivation

GI tools for non-functional properties?

RQ1 Availability — *Can we find them?*

RQ2 Usability — *Can they run?*

RQ3 Generalisability — *Can we recommend them?*

$\rightarrow$ **literature review**

$\rightarrow$ **experimental study**

Existing GI Surveys

Petke et al. (2018)[1]

- Genetic Improvement of Software: A Comprehensive Survey
- IEEE Transactions on Evolutionary Computation 22, 3
- 66 GI core papers (1995–2015)

Living Survey on GI[2]

- Based on Bill Langdon's GPBIB
- 468 GI-related papers (1985–2022)

[1] `https://doi.org/10.1109/TEVC.2017.2693219`
[2] `https://geneticimprovementofsoftware.com/learn/survey`

Survey Results

Methodology

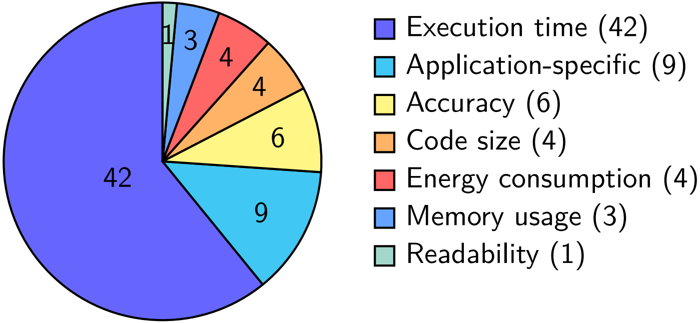
- Paper should focus on NFP
- Paper should propose, implement, or reuse a GI tool
- Paper should include experimental results

Literature review

Source	Dates	Papers	On NFP	With code
Petke et al. (2018)	(1995–2016)	66	27	19
Living survey on GI	(2016–2022)	264	63	45
ACM Digital Library	(2016–2022)	35	15	4
IEEE Xplore	(2016–2022)	57	10	9

RQ1: 63 unique relevant GI papers on NFP

GI of NFP in Practice



$\rightarrow$ **execution time is the most targeted NFP**

GI Tools for NFP

The quest for source code...

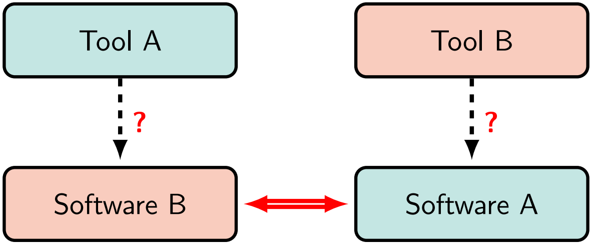
- 63 GI papers on NFP with empirical results
- only 31 with available code
- *only* 13 distinct tools

Validation

- No application-specific NFP (2 excluded: unnamed)
- No hard hardware requirement (1 fail: GEVO)
- Dependencies should be available (1 fail: Optmizer)
- Tools should run with provided examples (1 fail: HOMI)

RQ2: 13 distinct GI tools for NFP; 8 that we could run: GGGP, Gin, GISMO, locoGP, PowerGauge, PyGGI (+2 unnamed)

Tool Cross-Evaluation



Methodology

We test every tool on a new software, using an experimental setup lifted from a previous work involving a different tool (but same NFP).

Empirical Study

Gin

- Tested on SAT4J $\rightarrow$ OK

LocoGP

- Tested on GSON $\rightarrow$ gave up
- Far too much manual work

GISMO

- Tested on RNAfold $\rightarrow$ fail
- Unable to generate BNF grammar

GGGP

- Tested on MiniSAT $\rightarrow$ fail
- Unable to modify example

PyGGI

- Tested on GSON $\rightarrow$ OK

Unnamed tool (shader)

- Tested on MiniSAT $\rightarrow$ fail
- Designed to only work with shaders

Unnamed tool (OpenCV)

- Tested on MiniSAT $\rightarrow$ fail
- Unable to expose deep parameters

PowerGauge

- Tested on MiniSAT $\rightarrow$ fail
- Designed for assembly pipelines

Conclusion

RQ1 (Availability) 63 unique GI papers on NFP (mainly execution time)
RQ2 (Usability) 8 GI tools we could easily run
RQ3 (Generalisability) 2 GI tools we could easily reuse (Gin, PyGGI)

Observations:

- Poor availability
- Poor documentation
- Poor reusability
- (Public) GI tools are not industry-ready

Take-home message: **Release better (documented) code!**

Selected References

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.