Synthetic Benchmarks for Genetic Improvement

Aymeric Blot Justyna Petke

University College London, UK
UK EPSRC grant EP/P023991/1

GI@ICSE — 3 July 2020

In a Nutshell

Motivation:

- Empirical comparisons of GI approaches
- Parameter configuration of GI
- Genetic improvement of GI
- Quick experimentation for GI ideas

Idea:

- Premise: GI applied on software is very slow
- Bottleneck: fitness evaluation
- Proposition: synthetic benchmarks

Synthetic Benchmarks

Issues with real-world benchmarks:

- Evaluation is expensive
- Good data is scarce
- Uncertain features

Possible solutions:

- Surrogate modelling
- Artificial instances
- Synthetic benchmarks


Dang et al., GECCO 2017 (AC(AC) using surrogate modelling)


Malitsky et al., LION 2016 (Structure preserving instance generation)

Formalism

Standard GI:

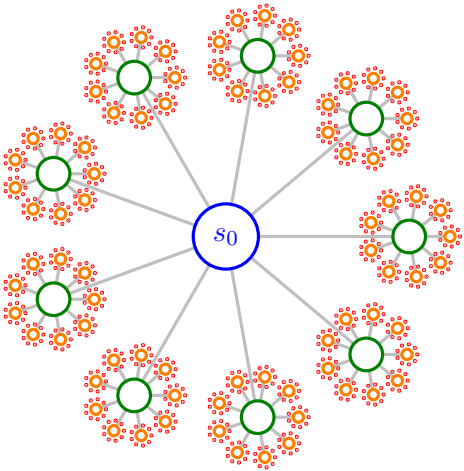
$$\text{(GI)} \quad \begin{cases} \text{optimise} & E[o(s,i),\ i \in \mathcal{D}] \\ \text{subject to} & s \in S \end{cases}$$

with:

- E: statistical population parameter (e.g., average)
- o: cost metric (e.g., running time)
- $\mathcal{D}$: input distribution (e.g., test cases, instances)
- s: software variants
- S: search space

Idea: Replacing $E[o(s,i), i \in (D)]$ by a single instantaneous query

Software Analysis



Search space:

- Around n deletions
- Around n^2 replacements
- Around n^2 insertions
- $\rightsquigarrow$ $\sum_{i=1}^{k}(n^{2i})$ sequences up to size k
- that's too big!

Assumption:

- Edits are independent
- $\rightsquigarrow$ only around n^2 fitness values
- reasonable to model

Synthetic Model

Empirical analysis:

- Sample edits
- Collect data, e.g.:
  - did it compile?
  - did it run?
  - was it correct?
  - how much better/worse?
- Compute underlying distribution

Contribution aggregation:

- Compilation errors propagate
- Runtime errors propagate
- Wrong outputs propagate
- Duplicate edits are ignored
- Fitness ratios are multiplied

E.g.: $[80\%, 100\%, 105\%] \rightarrow 84\%$

Conclusion

Problem:

- `GI(software)` is much slower than `software`
- `GI(GI(software))` is *much* much slower than `GI(software)`

Idea:

- Replace `software` with `model`
- `model` is free
- `GI(model)` is cheap
- `GI(GI(model))` should be reasonable

Advantages:

- Cheap, reusable benchmarks
- Model as complex as designed
- Possible focus on particular software feature

Selected References

Nguyen Dang, Leslie Pérez Cáceres, Patrick De Causmaecker, and Thomas Stützle.
Configuring irace using surrogate configuration benchmarks.
In Peter A. N. Bosman, editor, *Proceedings of the 12th Genetic and Evolutionary Computation Conference (GECCO 2017), Berlin, Germany*, pages 243–250. ACM, 2017.

Yuri Malitsky, Marius Merschformann, Barry O'Sullivan, and Kevin Tierney.
Structure-preserving instance generation.
In Paola Festa, Meinolf Sellmann, and Joaquin Vanschoren, editors, *Proceedings of the 10th International Conference on Learning and Intelligent Optimization, Revised Selected Papers (LION 10), Ischia, Italy*, volume 10079 of *Lecture Notes in Computer Science*, pages 123–140. Springer, 2016.