Genetic Engine: Grammar-Guided Genetic Programming without the grammar

Leon Ingelse, Guilherme Espada, Paulo Santos, Pedro Barbosa, Alcides Fonseca

LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal

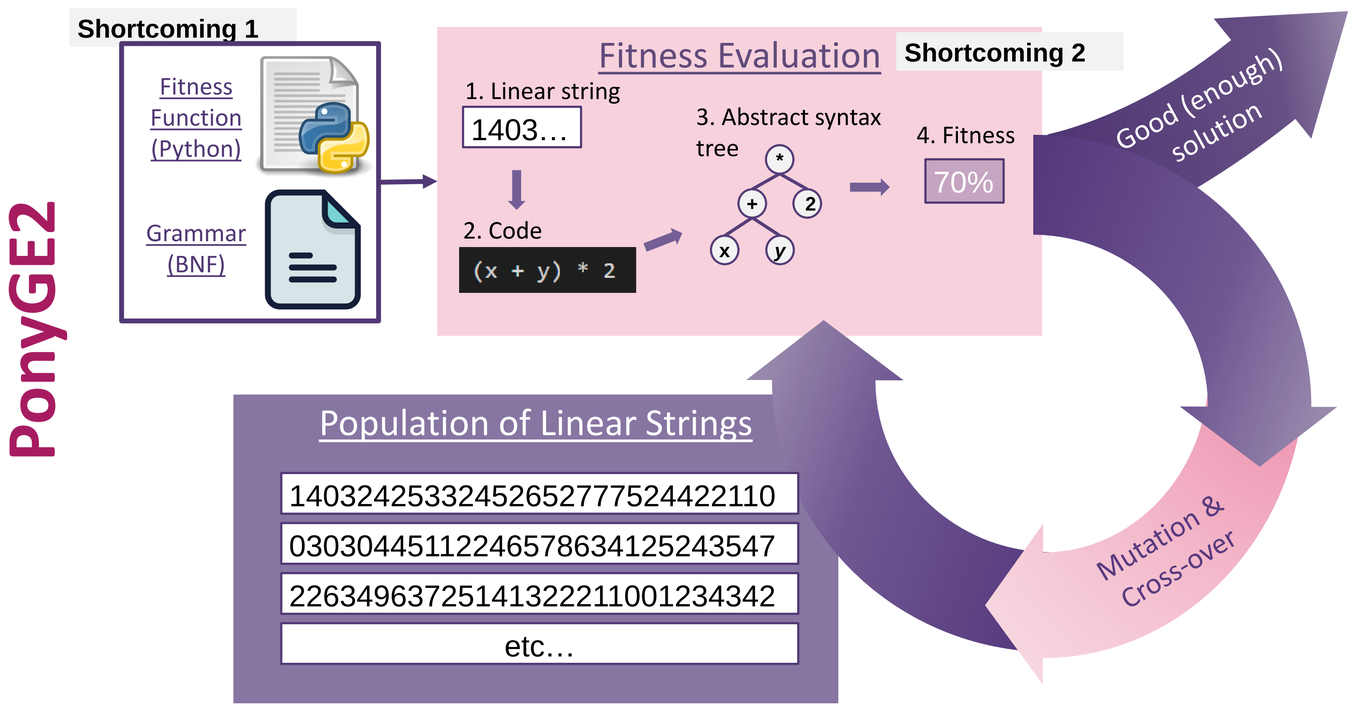
reliable software systems

Motivation

Genetic Programming (GP), a nature-inspired Machine Learning (ML) method, is praised for its ability to produce solutions from vast solution spaces. Grammar-Guided GP (GGGP) uses grammars to restrict the solution space, avoiding the exploration of solutions known to be invalid, as well as allows the user to design an interpretable domain language. This contrasts with popular ML approaches like Deep Neural Networks, which are far from interpretable.

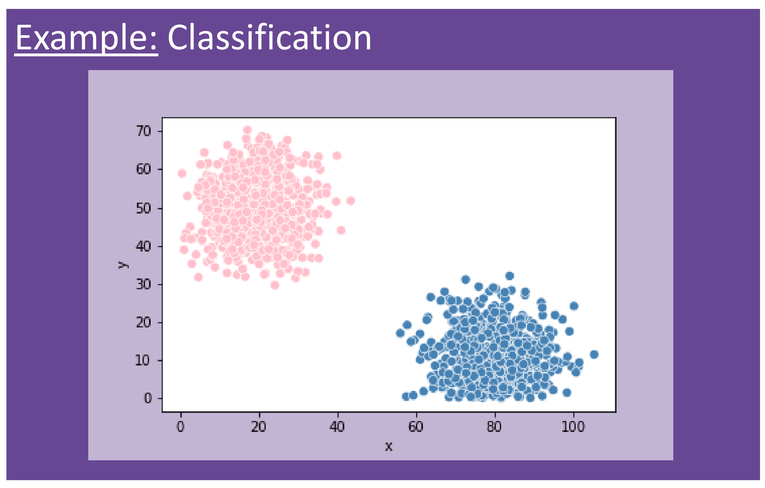
The current GGGP state of the art tool, PonyGE2, has two main **shortcomings**:

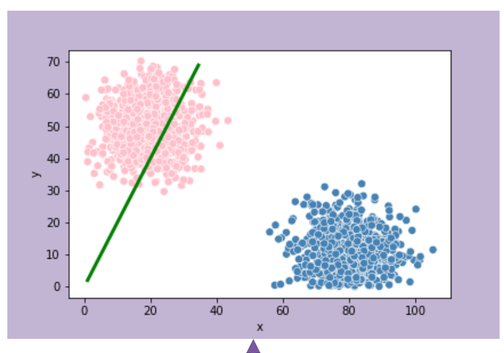
1. The grammar design is specified in a mix of BNF and the target language (Python), which does not support IDE features like autocompletion, linting, and type-checking.
2. Programs are unnecessarily converted from derivation trees to a textual representation and back to derivation trees during parsing.



Genetic Engine

We propose Genetic Engine, a pure Python GGGP framework that can encode solutions both as abstract trees, also known as derivation trees, as well as using the linear string approach of PonyGE.



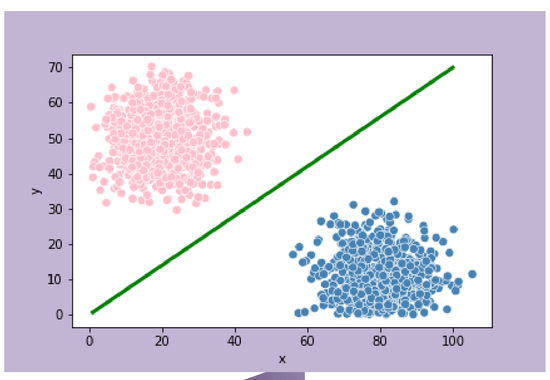


Accuracy: 70%

Direct Evaluation: Genetic Engine allows the use of derivation trees as individual representations. As such, it can directly evaluate individuals, resulting in performance advantages.

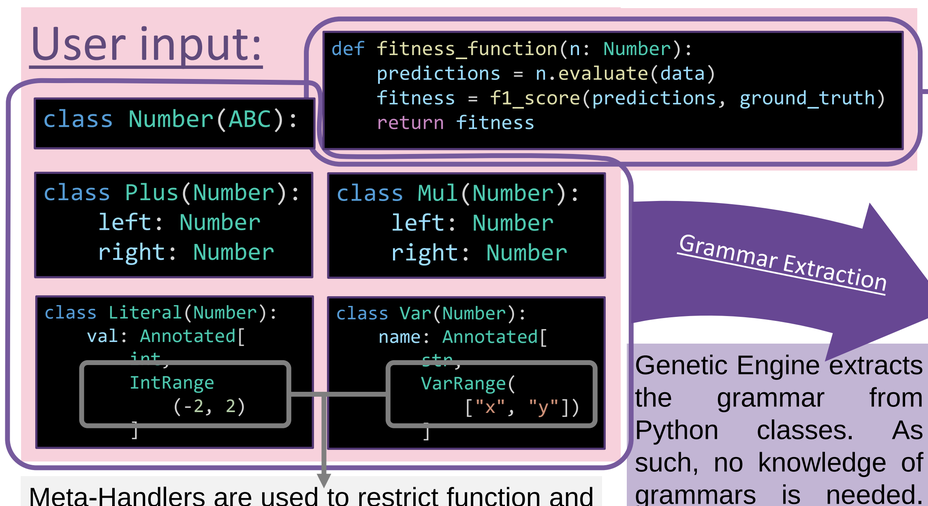
(x * -2) + y

Individuals can be converted to code



When the algorithm has finished or a pre-defined fitness was reached, the model returns the best individual.

Good (enough) solution



Meta-Handlers are used to restrict function and terminal generation in many ways. Here, they are used to restrict terminal values. Meta-Handlers can also be used to specify the generation probability of functions and terminals! This is also possible by BNFs, but only in a very cumbersome way that does not scale well.

Grammar Extraction

Genetic Engine extracts the grammar from Python classes. As such, no knowledge of grammars is needed. Furthermore, Genetic Engine is Python native, allowing the user to use Python libraries and legacy code, linting, and code completion.

Grammar:

```
Number   ::= Number + Number |
             Number * Number |
             Var | Literal
Var      ::= x | y
Literal  ::= -2 | -1 |
             0 | 1 | 2
```

Fitness Evaluation

Individuals are evaluated on their effectiveness at solving the problem at hand.

Cross-over & Mutation

The best individuals are transformed to obtain new individuals

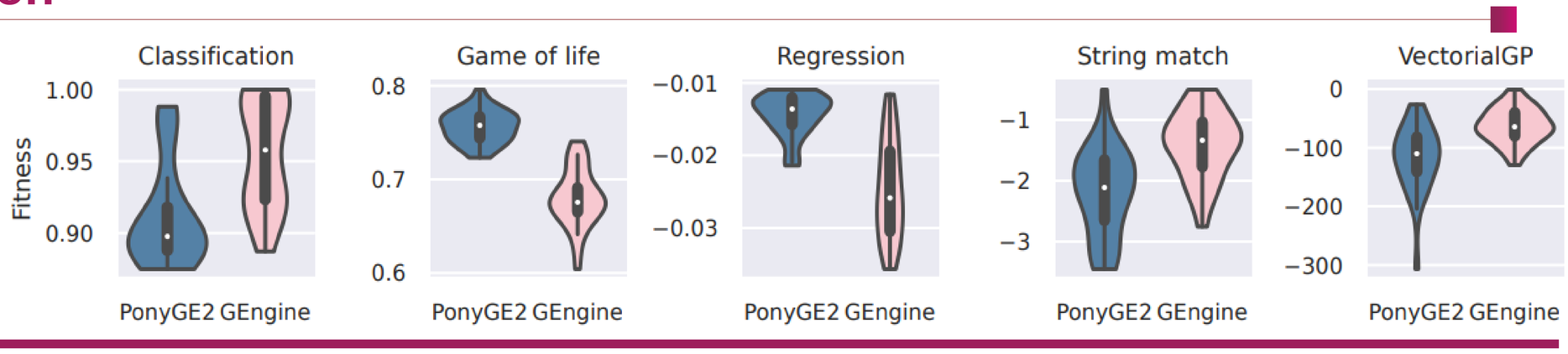
Individual initialisation

The initial population is initialized with randomly synthesized individuals.

Performance evaluation

To evaluate the performance of Genetic Engine, we compared it with PonyGE2 on 5 benchmarks. A higher fitness is better.

Genetic Engine shows to perform on par with PonyGE2, but it is more expressive due to Meta-Handlers. Furthermore, the ergonomic of Genetic Engine is higher, as it does not require BNF knowledge, and all Python tools can be applied directly.



Acknowledgements: This work was supported by the Fundação para a Ciência e a Tecnologia (FCT) under LASIGE Research Unit UIDB/00408/2020 and the *CONFIDENT* project (PTDC/EEI-CTP/4503/2014), and the CAMELOT project of Carnegie Mellon Portugal, and the RAP project under the reference (EXPL/CCI-COM/1306/2021).