Evolutionary Arithmetic Snippet Generation with Mutation and Selection

Emilio Zangger
Computational Algorithm Lab, Melbourne, Australia

Abstract

We present an evolutionary framework for generating and optimizing arithmetic code snippets. Each snippet consists of two integers (1–10) combined with operators {+, -, *, //, **, %}. An evolutionary algorithm iteratively selects top-performing snippets based on evaluated results, applies mutations to components (operands or operator), and produces successive generations. Safe evaluation is used to handle division, modulo, exponentiation, and potential errors. This framework demonstrates stochastic optimization in a lightweight, interpretable, and fully automated code environment.

Introduction

Evolutionary computation and genetic programming have been applied extensively to automatic program synthesis and numerical optimization. In this study, we implement a minimal evolutionary system focused on arithmetic code snippets. Unlike previous work focusing on large programs, we isolate the principle of variation, evaluation, and selection in a small, interpretable domain.

Methods

Snippet Generation: Random snippets are generated in the format ' ' with operators including +, -, *, //, **, %. Safe Evaluation: Each snippet is evaluated using a function that handles division by zero, overflow, and invalid operations, assigning a score of zero when errors occur. Mutation: Randomly modifies one component of a snippet (operand or operator) to produce variation. Evolutionary Algorithm: Initialize a population of 20 random snippets, evaluate, select top 5, mutate top 3 to fill next generation, repeat for 5 generations, track highest-scoring snippets.

Results

A typical run demonstrates rapid convergence toward high-value arithmetic expressions, improvements in average and maximum scores across generations, and effective exploration of solution space via mutation. Example final top snippets may include: '9 ** 2 -> 81', '7 * 8 -> 56', '10 + 10 -> 20'.

Discussion

This framework illustrates key principles of evolutionary computation: stochastic variation, selection pressure, and iterative improvement. Constraining the domain to simple arithmetic and adding safe evaluation keeps it interpretable while demonstrating non-trivial optimization dynamics. Applying evolutionary methods to dynamic code snippets with mutation and error-safe evaluation in a minimal domain is a novel pedagogical demonstration. Potential extensions include larger operand ranges, multi-step expressions, and convergence visualization.

Conclusion

The evolutionary snippet generator demonstrates a compact yet illustrative system for automated stochastic optimization. By combining random initialization, safe evaluation, mutation, and selection, it provides an accessible model of evolutionary computation for code generation in a controlled, interpretable environment.

References

Koza, J. R. (1992). Genetic Programming: On the Programming of Computers by Means of Natural Selection. Banzhaf, W., et al. (1998). Genetic Programming: An Introduction. Holland, J. H. (1992). Adaptation in Natural and Artificial Systems.