Improving Evolutionary Algorithms for Automatic Software Development

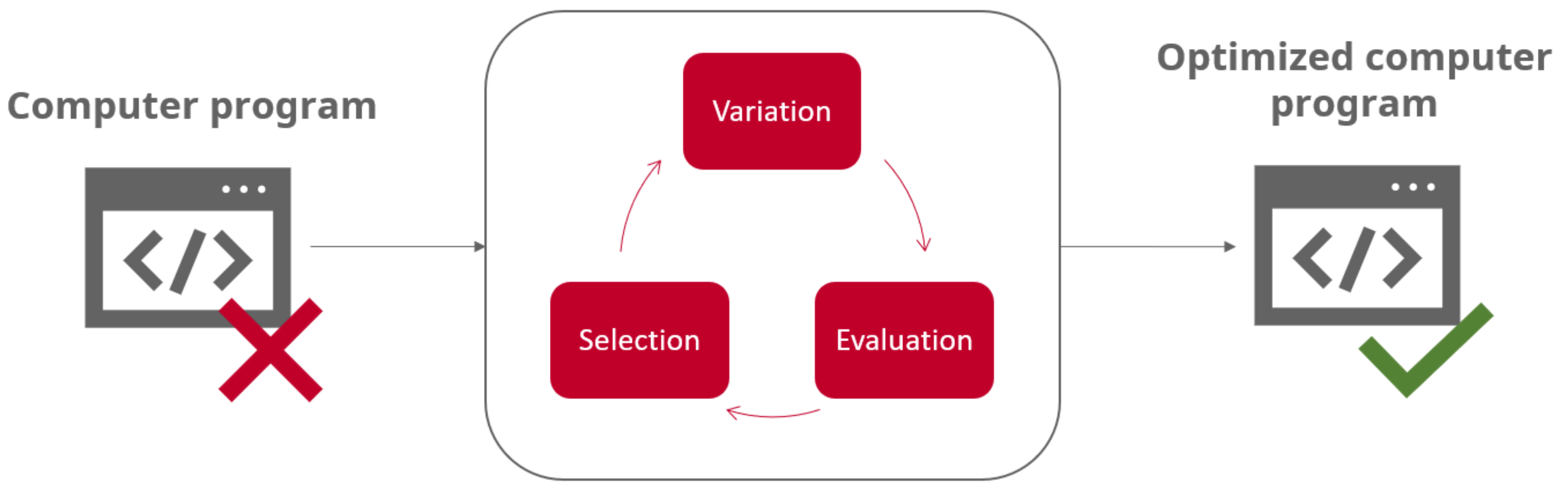
Alina Geiger

Chair of Information Systems and Business Administration, Prof. Dr. Franz Rothlauf, JGU Mainz

GUTENBERG ACADEMY **FELLOWS** PROGRAM

Evolutionary Algorithms (EAs)

In the field of artificial intelligence, EAs are methods for solving optimisation problems. With EAs, we are able to find an (approximately) optimal solution to a problem from a large number of possible solutions. EAs solve optimisation problems by imitating the natural evolutionary process.



The evolutionary process starts with a population of random computer programs that are evaluated using test examples. Through selection and variation, the population of computer programs evolves over many generations until an (approximately) optimal solution is found.

Areas of application include:

- Medicine Development of computer programs based on health data that support medical professionals [6].
- Finance Solving optimisation problems, e.g. forecasts for stock market prices [1].
- Software development Automated repair of bugs in program code or automated optimisation of software runtime [4, 7].

Research Aim

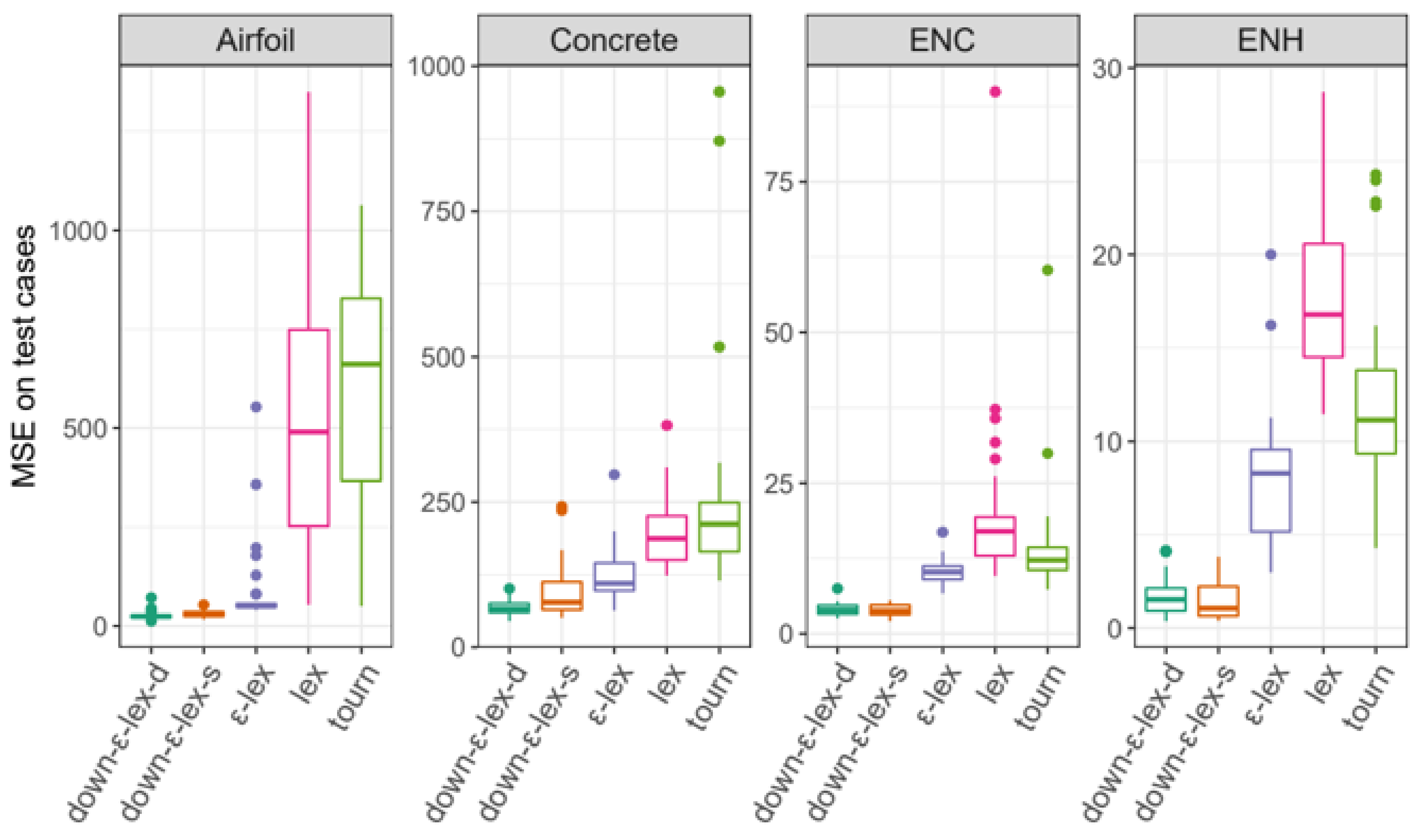
How can we further improve EAs in order to develop software automatically?

- Selection How do we decide which computer programs will be the parents for the next generation?
- Variation How can we introduce new program code into our population of computer programs?
- Evaluation How do we evaluate the performance of the computer programs in our population?

The aim of my thesis is to improve state-of-the-art EAs in order to expand the possibilities for the automatic generation and maintenance of software.

Improving the Selection Process

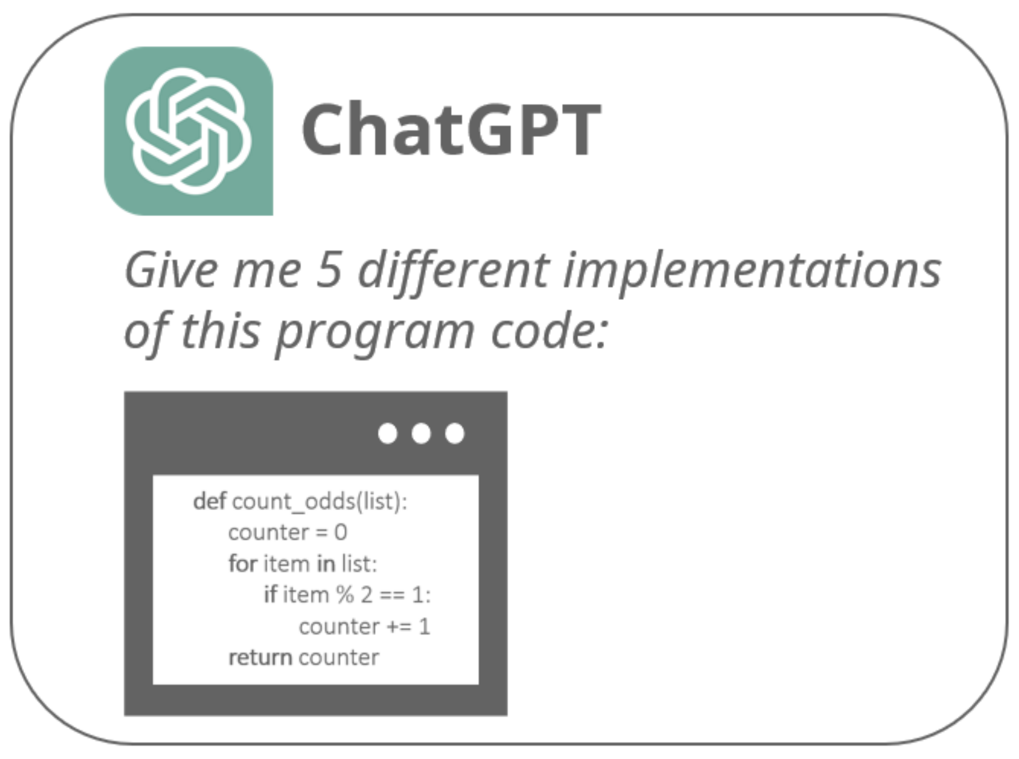
The selection process determines which computer programs survive and act as parents for the next generation. In our work [3], we introduced a novel selection method called down-sampled ϵ-lexicase selection (down-ϵ-lex) and tested its performance against other selection methods on several benchmark problems.



With our selection method, we were able to find computer programs of higher quality (with smaller error in terms of the MSE) on the studied problems compared to other selection methods. Overall, the performance improvements using down-sampled ϵ-lexicase selection are up to 85% compared to ϵ-lexicase selection.

Improving the Variation Process

In our work [2] we investigated the use of ChatGPT as a variation operator with the aim to improve the runtime of software and to repair bugs. To do this, we asked ChatGPT for 5 different implementations of a code snippet.



Give me 5 different implementations of this program code:

```
def count_odds(list):
    counter = 0
    for item in list:
        if item % 2 == 1:
            counter += 1
    return counter
```

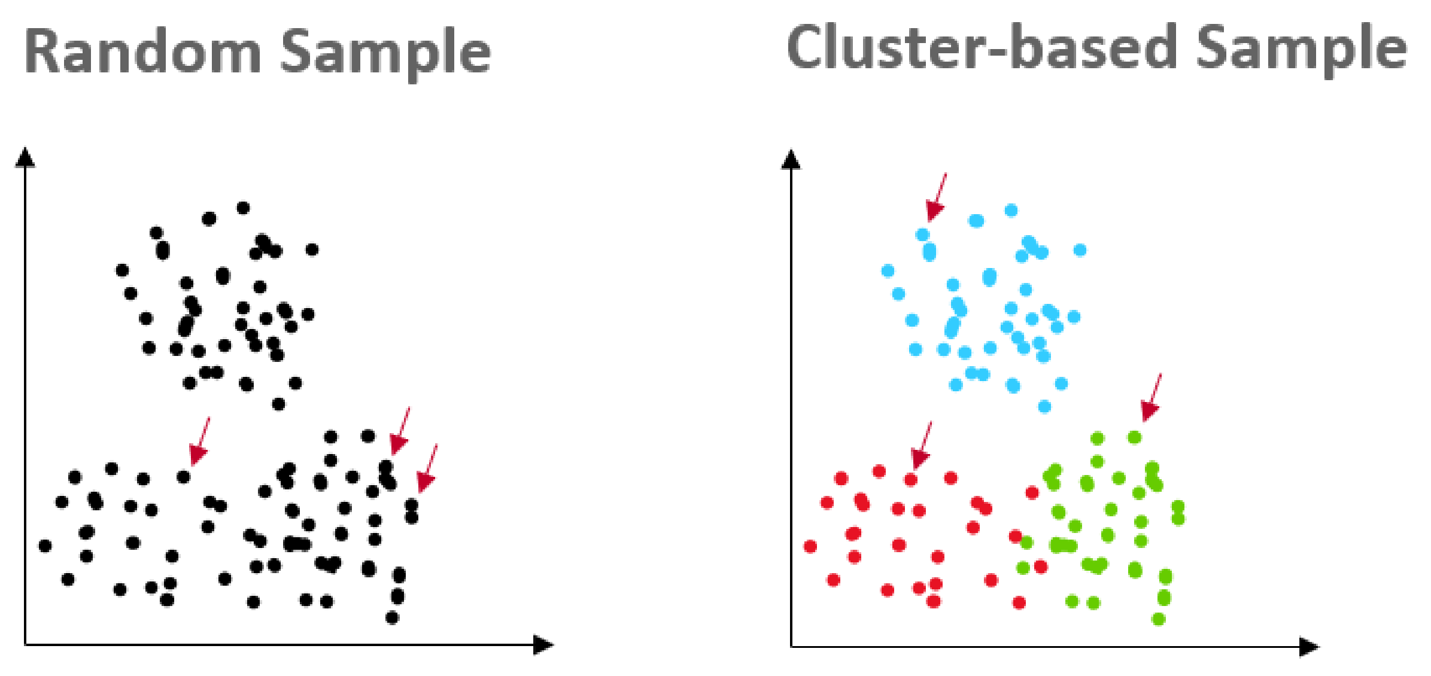
Main Results:

- The number of patches passing software tests is up to 75% higher when using ChatGPT as a variation operator compared to traditional operators.
- The patches generated by ChatGPT are less diverse.
- The patch with the best runtime improvement is found by a traditional variation operator.

Improving the Evaluation Process (Outlook)

During the evolutionary process, we need to evaluate the quality of the computer programs in our population in order to do the selection process. The quality of computer programs is usually evaluated based on their performance on test examples.

- Problem Evaluating computer programs using a large number of test examples is expensive.
- Proposed Solution Use a random subset of test examples to evaluate computer programs [5].
- Open Question How can we select a subset of test examples that tests as many behaviors of the computer program as possible?



References

[1] A. Brabazon, M. Kampouridis, and M. O'Neill. Applications of genetic programming to finance and economics: past, present, future. *Genetic Programming and Evolvable Machines*, 21:33–53, 2020.

[2] A. E. I. Brownlee, J. Callan, K. Even-Mendoza, A. Geiger, C. Hanna, J. Petke, F. Sarro, and D. Sobania. Enhancing genetic improvement mutations using large language models. In *Search-Based Software Engineering*, pages 153–159. Springer Nature Switzerland, 2024.

[3] A. Geiger, D. Sobania, and F. Rothlauf. Down-sampled epsilon-lexicase selection for real-world symbolic regression problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23, page 1109–1117. ACM, 2023.

[4] S. O. Haraldsson, J. R. Woodward, A. E. Brownlee, and K. Siggeirsdottir. Fixing bugs in your sleep: How genetic improvement became an overnight success. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1513–1520, 2017.

[5] J. G. Hernandez, A. Lalejini, E. Dolson, and C. Ofria. Random subsampling improves performance in lexicase selection. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '19, pages 2028–2031. ACM, 2019.

[6] W. G. La Cava, P. C. Lee, I. Ajmal, X. Ding, P. Solanki, J. B. Cohen, J. H. Moore, and D. S. Herman. A flexible symbolic regression method for constructing interpretable clinical prediction models. *NPJ Digital Medicine*, 6(1):107, 2023.

[7] J. Petke, S. O. Haraldsson, M. Harman, W. B. Langdon, D. R. White, and J. R. Woodward. Genetic improvement of software: a comprehensive survey. *IEEE Transactions on Evolutionary Computation*, 22(3):415–432, 2017.