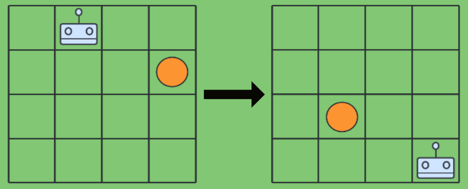
Evolving design patterns for program synthesis

1. Background

Program Synthesis (PS):

- The task of creating programs from a specification.
- In Inductive Program Synthesis specification comes in the form of input/output examples [1].
- Examples:



Robot planning domain

```
“animals.jpg”  ⟶ “jpg”
“MyReport.pdf” ⟶ “pdf”
“website.html” ⟶ “html”
```

String transformation domain

- Generated programs:

```
[MoveRight,MoveRight,MoveDown,Grab,
MoveLeft,MoveLeft,MoveDown,Drop,
MoveRight,MoveRight,MoveDown]
```

```
[LoopWhile(IsLetter,
[MoveRight]),Drop]
```

Design Patterns:

- Useful abstractions that can be reused across programs.
- Example:

```
DoubleAndDown(Expression) = [Expression, Expression, MoveDown]
```

- Reduce number of expressions needed for a task.
- Challenge: how to come up with them?

Genetic Algorithms (GA):

- Search heuristic for finding an optimal solution.
- Inspired by natural selection: survival of the fittest.
- Evolutionary operators: selection, crossover, mutation.

2. Research Question

Can Design Patterns evolved with Genetic Algorithms increase the accuracy of Program Synthesis?

3. Method

- Literature search
- Design pattern implementation and integration
- Genetic algorithm design and implementation
- Evaluation

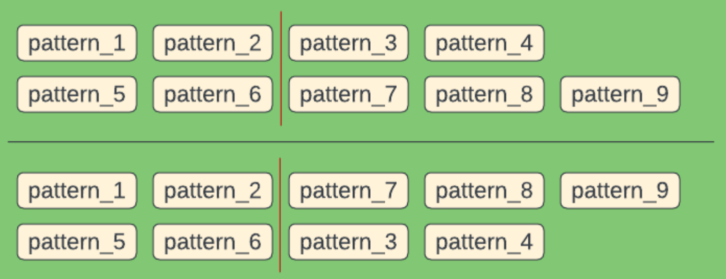
4. GA implementation

- Chromosome encoding: list of design patterns.
- Generate a random population of chromosomes.
- How to measure the performance of a chromosome?

$$fitness = c_{avg} \cdot \frac{1}{t_{avg}}$$

Average of correct examples per task   Average execution time

- Elitism: selecting a fraction of best performing individuals.
- Fitness-proportionate selection: roulette wheel sampling.
- Single-point crossover swaps the subparts at random point:



pattern_1 pattern_2 | pattern_3 pattern_4

pattern_5 pattern_6 | pattern_7 pattern_8 pattern_9

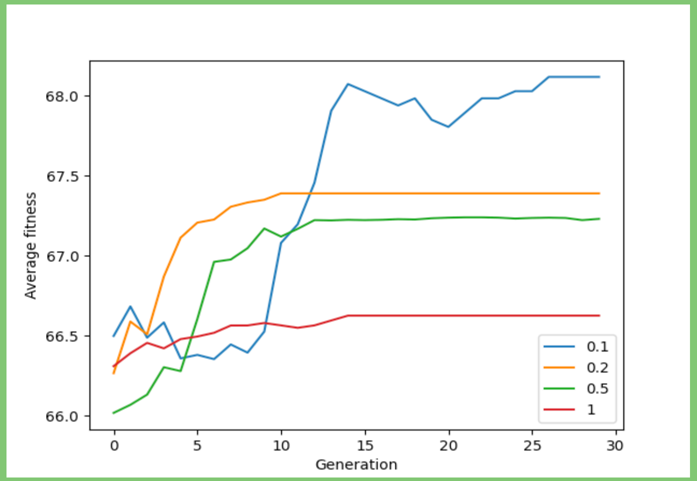
pattern_1 pattern_2 | pattern_7 pattern_8 pattern_9

pattern_5 pattern_6 | pattern_3 pattern_4

- Mutation replaces a random design pattern.
- Resulting offspring are forwarded to the next generation.
- GA terminates when a number of generations was reached.

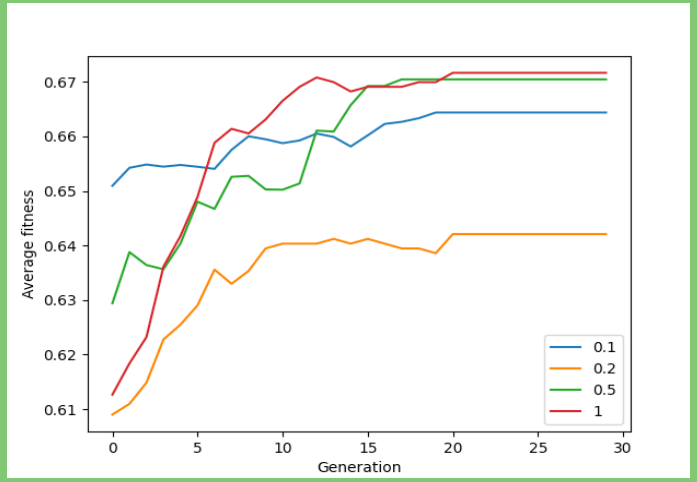
5. Results

- Robot planning domain:



  - Comparing performance to using no design patterns: same average accuracy but lower average execution times.

- String transformation domain:



  - Comparing performance to using no design patterns: worse average accuracy and average execution times.

7. Conclusions

1. The accuracy was not improved by the evolved design patterns.
2. Comparison to other heuristics needed.
3. Best patterns according to fitness were always appearing in the very first generation - possible problem in integrating them with the invented tokens.

TU Delft

Fabian Radomski

Supervisor: Sebastijan Dumančić

References:

[1] Armando Solar-Lezama. Introduction to program synthesis, 2018.