Genetic Algorithm for Evolving an Objective Function of a Program Synthesizer

Student: Nikolaos Efthymiou || Supervisor: Sebastijan Dumančić, Algorithmics group – TU Delft

TU Delft

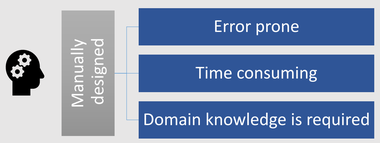
1. Introduction

- **Program Synthesis (PS) problem:** *Given a programming language that defines the program space and a user intent/specification, find a program that satisfies this specification.*

Elements of PS

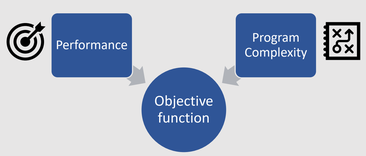
User specification	Program Space	Search technique
I/O examples (Programming By Example)	Domain-Specific Language (DSL)	Brute synthesizer [1]

***Objective function**: key aspect of a program synthesizer to guide the search*



A Genetic Algorithm can partially automate the design of an objective function.

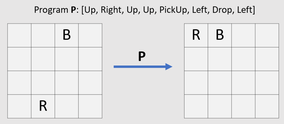
General criteria



Well-studied PS domains (String transformation and Robot Planning) [1]

Input	Output
22 July,1983 (35 years old)	JUL
30 October,1955 (63 years old)	OCT
2 November,1954 (64 years old)	NOV

Program **P**: [Up, Right, Up, Up, PickUp, Left, Drop, Left]



- Manually-designed objective functions exist for the Robot and the String domains [2, 3].
- Genetic Algorithm (GA) is a well-known population-based metaheuristic algorithm
  - GAs have been used for the evolution of functions
  - Elements: chromosome representation, fitness function, selection, mutation and crossover
- A GA could take as input several user-defined *domain-specific local distance functions* and combine them to construct objective functions

'length difference', 'difference in number of upper-case characters', ...

$$\ll \quad \mathrm{OPT}(i,j) = \begin{cases} j & \text{if } i = 0 \\ \min \begin{cases} \mathrm{OPT}(i-1,j-1) & \text{if } x_i = y_j \\ 1+\mathrm{OPT}(i-1,j-1) & \text{if } x_i \equiv y_j \\ 1+\mathrm{OPT}(i-1,j) & \text{otherwise} \end{cases} & \\ \infty & \text{if } j = 0 \end{cases}$$

[3]

- **Research Question:** *How effective is a program synthesizer using an objective function that is evolved by means of a Genetic Algorithm?*

2. Methodology

- Objective functions involving user-defined domain-specific *local distance functions* are evolved with the *GeneticObjective* GA.

Local distance functions used in our experiments

$$\mathcal{F}_{\text{robot}} = \begin{Bmatrix} L^1(p_r, p_b^*), \\ L^1(p_r, p_r^*), \\ L^1(p_b^*, p_r^*), \\ L^1(p_r, p_b), \\ L^1(p_b, p_b^*) \end{Bmatrix} \quad \mathcal{F}_{\text{string}} = \begin{Bmatrix} abs(|s| - |s^*|), \\ min(|s|, |s^*|) - |s \cap s^*|, \\ abs(i - i^*), \\ abs(cU(s) - cU(s^*)), \\ abs(cL(s) - cL(s^*)) \end{Bmatrix}$$

Elements of GeneticObjective

Termination condition	Reaching a given number of generations
Fitness function*	$\mathcal{V}(T, \mathcal{E}_D) = w_1 S + w_2(1-U) + w_3(1-R)$
Chromosome encoding	Algebraic expressions in the form of binary expression trees
Initial population	Random expression trees of a given maximum height
Mutation	Replacement of a random node
Crossover	Exchange random subtrees of the parents
Parent selection	Deterministic k-tournament selection
Elitism	Mostly elite size = 6

*(S: percentage of solved tasks, U: average percentage of unsolved examples over the unsolved tasks, R: average normalized runtime of Brute)

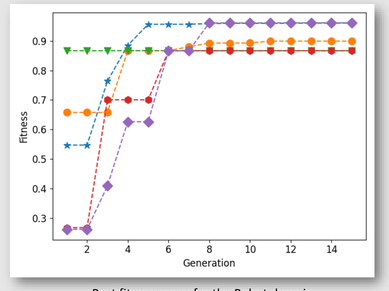
- Evaluation of *GeneticObjective*: experiments in the Robot and String domains, using the Brute synthesizer [1].
  - Analyse the convergence of *GeneticObjective*
  - Compare fitness value of best function found with the value obtained by the manually-designed objective function

3. Results

Domain	Manual fitness	Best fitness GA	Time taken
Robot	0.95	0.95	36m
String	0.24	0.29	98m

Robot Planning:

- Train set size=150, test set size=350
- Suboptimal solutions for p_m=0.01 (premature convergence) and p_m=0.16 (prevents convergence)
- Poor results for e=2
  - Low value => lack of exploitation of the good solutions
- The diversity of the population affects the performance of the GA: Poor performance for a low value of p_c=0.5
- There is only a single task => [w_1, w_2, w_3] = [0, 0.9, 0.1]



Best fitness curve for the Robot domain

String transformation:

- Train set size=150, test set size=225
- [w_1, w_2, w_3] = [0.6, 0.3, 0.1]
- Poor results with high p_m value (0.2) (it resembles a random search)
- Most configurations led to similar results

- H, p and G do not significantly affect the performance of GeneticObjective

4. Conclusions

- The experiments we conducted showed that our approach is effective in both domains, by reaching or even surpassing the effectiveness of the manually-designed function.
- Low scores in the String domain could be attributed to the inherent difficulty of the domain. Thus, a more effective synthesizer would be required to obtain better results.

5. Limitations and Future Work

- Limited scope regarding the number of configurations tested and the number of trials per configuration.
- An interesting direction: integrate an existing manually-designed objective function in the fitness function of *GeneticObjective*.
  - Consider the distance to the correct output, together with the percentage of examples/tasks solved and the running time
- Eliminate equivalent expressions
- Consider more complex objective functions, e.g. add conditional expressions.

References

1. Andrew Cropper and Sebastijan Dumancic. Learning large logic programs by going beyond entailment. ArXiv, abs/2004.09855, 2020.
2. Stef Rasing. Improving inductive program synthesis by using very large neighborhood search and variable-depth neighborhood search. https://repository.tudelft.nl/islandora/object/uuid:a24ed4f6-6abd-4661-86b8-c5a965d62e4e?collection=education, 2022.
3. Bas Jenneboer. Program synthesis with a*. https://repository.tudelft.nl/islandora/object/uuid:873c3b33-2501-4438-a610-6dcb8ab8ad72?collection=education, 2022.