How do we adjust the FrAngel program synthesiser to discover diverse subprograms?

Playing Games with Program Synthesis: Solving Minecraft

Author
Georgi Latsev
g.s.latsev@student.tudelft.nl

Responsible Professor
Sebastijan Dumančić
s.dumancic@tudelft.nl

Supervisor
Tilman Hinnerichs
t.r.hinnerichs@tudelft.nl

01 Introduction

Program synthesis is the process of automatically generating programs satisfying a given specification.

```
Num = 1
Num = x
Num = Num + Num
```

```
x = 5 → 7
x = 1 → 3
x = 0 → 2
```

```
x + 1 + 1
```

FrAngel - is a program synthesis algorithm that is based on two main concepts - fragments and angelic conditions. **Fragments** are parts of programs that were found to be useful in previous iterations of the algorithm. While **angelic conditions** are placeholders for the condition of control structures such as ifs and loops.

```
x + 8 * y
```

```
y
```

```
if (<angelic>)
```

```
x
```

```
8 * y
```

```
8
```

```
while (<angelic>)
```

Exploration - searches the unexplored space to find useful actions and programs.

Game rewards - the agent receives a reward based on the distance moved toward or away from the goal.

02 Research Questions

Q1: How do we define program synthesis from rewards?

Q2: How to explore game environments to discover useful actions?

Q3: How do we adjust the FrAngel program synthesizer to discover diverse subprograms?

03 Methodology

Generalise FrAngel

```
Num = 1 | 2
Num = Fragment_Num
Fragment_Num = 2
```

- Allow arbitrary iterators
- Allow any grammar

```
Statement = if Bool Statement end
if <Hole> Statement  end
```

Define program synthesis from rewards - the reward is split into different reward thresholds represented as the output of IO examples

```
[8, 16, 24, 32, 40, 48]
```

Integration with Minecraft environment - the algorithm is run with checkpointing, starting from the previous best position from the last cycle

Run experiments - experiment with various modifications for exploration on the dense navigation task in MineRL

04 Experiments

One of the experiments investigates how **varying the smallest reward checkpoints** affects the effectiveness of the FrAngel algorithm by altering the length of its pure exploration phase and subsequently the quality of the fragments utilized.

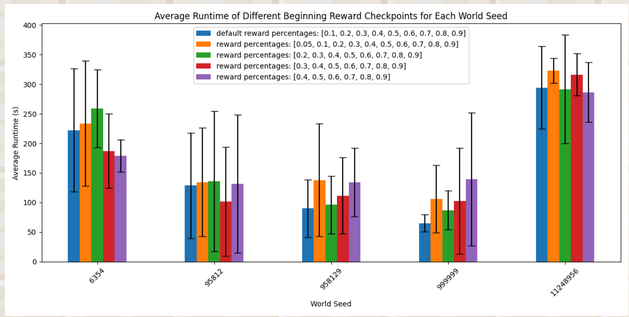

Fig 1: average runtime for each reward setup for each world

It was found that the **difficulty of the easiest output example should be balanced for the best performance - optimizing the exploration.**

Easier output examples could lead to less relevant fragments - that will guide the search in the wrong direction.

Harder output examples lead to more exploration before any fragments are found and exploited.

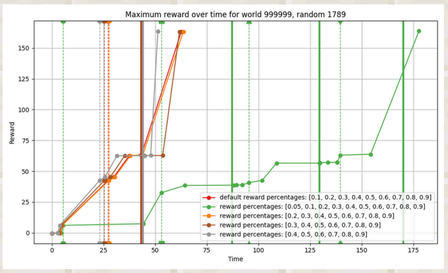

Fig 2a: maximum reward over time for a particular setup

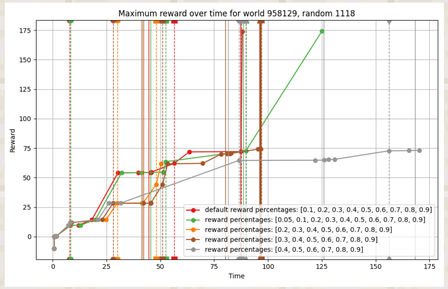

Fig 2b: maximum reward over time for a particular setup

05 Conclusion and Limitations

We introduced generalised FrAngel, defined program synthesis from rewards via input-output examples and tackled the navigation task in Minecraft.

The results are limited due to the low amount of runs for each setup compared to the many possible combinations of world and random seeds.