IGGI GD2 Lab Exercise: Rolling Horizon Evolution

IGGI GD2 Lab Exercise: Rolling Horizon Evolution 1
Introduction 1
Downloading and Running the Software (5 mins) 1
Asteroids and CaveSwing (30 mins) 1
Defeating the Default Agent (20 mins) 2
Experimenting with long reward horizons. 2
Agent Optimisation (1.5 hours) 2
Questions: 3
Summary 3

Introduction

The purpose of this lab is to gain experience of using the Rolling Horizon Evolution agent described in the lectures, and given in the codebase (see below).

The lab script should be doable within a couple of hours.

In the lab you will:

- Check out the latest software from the github repo
- Run a RHEA agent to player Asteroids and CaveSwing
- Play with the CaveSwing game to see where the default RHEA agents succeeds and fails
- Set up a search space for a RHEA agent for CaveSwing, and use the NTBEA to optimise the RHEA agent.

Downloading and Running the Software (5 mins)

Download the software from the following Github:
https://github.com/ljialin/SimpleAsteroids

(we suggest you download the .zip file)

Open the project in your favourite Java IDE (we recommend Intellij Idea).

Asteroids and CaveSwing (30 mins)

Start by running the file: asteroids.GameSpeedTest with its default settings.

You should see the agent play reasonably well (let's define that as achieving average scores in excess of 8,000).

In the main method you will see the lines:

```
Game.seqLength = 100;
Game.nEvals = 20;
```

Experiment with different numbers of sequence evaluations (also referred to as rollouts), and with different sequence lengths. The product of these values gives the number of game ticks used per decision. As you experiment with the numbers, there is no need to keep this product constant. For example, while keeping the same sequence length, how small can the nEvals be before performance steeply declines? (e.g. average score below 5,000).

Repeat the experiment for CaveSwing, by running caveswing.EvoAgentVisTest.

The default agent should play ok with these settings:

```
CaveSwingParams params = new CaveSwingParams();
params.maxTicks = 2000;

// todo: how does changing the parameter settings affect AI agent performance?
// todo: Can you settings that make it really tough for the AI?
params.gravity.y = 0.2;
params.gravity.x = -0.0;
params.hooke = 0.01;
```

Defeating the Default Agent (20 mins)

Continuing on from the above, experiment with the game parameters (by setting them as shown above, but to different values and also tuning additional parameters such as nAnchors). Can you find versions of the game playable that you can play, but for which the default AI fails?

Experimenting with long reward horizons.

Now set all the score related parameters to zero, apart from successBonus (leave this at its default value of 1000). Does the agent now fail where it previously succeeded? Try to find instances where the default rolling horizon player can still outperform the random player, despite the delayed reward.

Agent Optimisation (1.5 hours)

Now optimise an agent to play CaveSwing.

To do this, look again at the code for yesterday's lab which optimised the game parameters (hyperopt.TuneCaveSwingParams)

Now do a refactor/copy of that in to the class hyperopt.TuneCaveSwingAgent.

You'll need an AnnotatedFitnessSpace for this: you'll find an incomplete one in the file: caveswing.design.EvoAgentSearchSpaceCaveSwing

Towards the end of the evaluate method you'll find where some code needs to be added:

```
// todo now run a game and return the result
// use the evaluation code from yesterdays lab to evaluate the agent we already
made
double value = Math.random();
logger.log(value, x, false);
return value;
```

By studying the code you used yesterday for evolving the game design, you may be able to figure this out, but please ask if stuck.

When you have it running, now run some experiments by modifying hyperopt.TuneCaveSwingAgent to call the new AnnotadedFitnessLandscape.

When you get the optimiser working, study the outputs of each run.

Questions:

- What are the optimal values for sequence length?
- What is the impact of turning off the shift buffer?

The answers you get from the model have to be interpreted with caution. Why?

For the best solution you can find, test it with and without the shift buffer. What impact on performance to you observe now?

Summary

We have now covered how to use the codebase to optimise both the game design, and an agent to play the game. These are useful tools that you can use to optimise many different systems.