Chapter 8 - GVGAI without VGDL

Exercises

All the code is available in a Github repository[1]. Use the checkpoint[2] in order to run the same versions presented in this chapter[3].

1 Running the Demos

The first exercise is to download the code and run the following demos. We recommend doing this in a Java IDE such as Intellij IDEA or Eclipse. This enables the code to be easily modified and re-run. Running the demos below will should take a around 5 minutes. If you spend time adjusting the agent's parameters then this could take 30 minutes or more.

Each demo shows the same Rolling Horizon Evolution Agent playing the game. Note how well the agent is able to play each one without any modification.

1.1 Asteroids

Running the following class will run the default RHEA agent.
`asteroids.RunAsteroidsDemo`

1.2 Planet Wars

Run: `spinbattle.actuator.SourceTargetActuatorTest`. It will play the standard RHEA agent (Player 1, yellow) against a hand-coded heuristic agent (PLayer 2, Blue). You can also play against any of the agents. For example, running `spinbattle.test.HumanInterfaceTest` pits you against a heuristic player. You play as they yellow player. To play the game you click a source planet (must be owned by you, hence yellow) followed by the target planet you wish to invade (if blue or grey) or support if already owned by you (yellow). The interface is a bit basic, in that clicking the source planet has no discernible effect until the target planet is subsequently clicked.

[1] https://github.com/ljialin/SimpleAsteroids
[2] https://github.com/ljialin/SimpleAsteroids/commit/6ff5e7a8881be1e6d491edb527dd3bdcd9489b25
[3] These exercises are also available at this book's website: `https://gaigresearch.github.io/gvgaibook/`

1.3 Cave Swing

Run: `caveswing.test.EvoAgentVisTest` to test the default RHEA agent. To play the game, run: `caveswing.test.KeyPlayerTest`.

2 RHEA Exercise

All the above demos use the default RHEA agent with the same parameters. Change each one from its default value and observe the effects.

- `useShiftBuffer`: The default is `true`. What is the effect of changing it to `false`?
- `sequenceLength`: The default is 100. What are the effects of much lower values such as 10 on each game?
- `nEvals`: The default is 20. Explore how low can you go before performance significantly degrades?

3 Agent Comparison Exercise

MCTS is much more widely used than RHEA, although in our experience RHEA is simpler to implement and often provides better performance. The exercise is to make some of your own comparisons. Start by running this test on planet wars that runs a round-robin league: `spinbattle.league.RoundRobinLeagueTest`. If using the exact checkpoint given you will see that some of the possible players have been commented out. Include the MCTS player and see how it compares to the RHEA player. Set the total number of games high enough to obtain statistically significant results. This will typically take between 5 minutes to 1 hour depending on the number of games chosen, the number of players in the league and the simulation budget allowed for each of the SFP players.

4 Game Tuning Exercise

We can put the speed to good use by running some efficient tuning of the game parameters. A starting point for exploring this is provided. Begin by studying and running: `hyperopt.TuneCaveSwingParams`. This takes around 10 seconds for one complete optimisation run. It uses the NTBEA to perform an efficient optimisation of the game's parameters. Observe the output of the run, which provides detailed statistics on the effects of various parameter combinations.

Within the output, you will find the optimised solution encoded as an array of int, together with a listing of each parameter value. Try copying these values back in to the `KeyPlayerTest` code (see examples in method `getSearchSpaceParams` of `KeyPlayerTest` and ensure that the call to it on line 21 is uncommented) and play through them.

By default the optimisation is set up to maximise the score difference between a mediocre RHEA agent and a smart RHEA agent. This often produces fiendishly difficult levels for human play. As an exercise, change the "dumb" player in `caveswing.design.CaveSwingFitnessSpace` to be a random player (lines 140 - 142), and observe the effects on the evolved games. You may find them a bit easier to play now!