

Problem Set 6.3: Due April 7

For this assignment, you will extend your Soar agent for the Eaters world. You should turn in your Soar code and any supporting materials including a readme file that describes what you implemented and how it works.

1. (10 points) Create an agent that intensifier's the height and width of the map by exploring it. Once the size is identified, the agent should print the message *The map is h by w* where h and w are replaced by the height and width respectively. The agent should function correctly on the maps *circle*, *random*, *jump* and *small*.
2. (10 points) Create an agent that uses subgoals to distinguish between three high level strategies: eat (where your normal strategy is pursued), flee (where the eater moves away from another eater whose score is less than its own), follow (where the eater tries to collide with an eater whose score is greater than its own). Both flee and follow should only be in effect if another eater is observer on the input link. You can also add other goals so as to implement all the features for this assignment in the same agent.
3. (10 points) Modify your eater so it can determine if another agent is also on the map. If it identifies a multi-agent setting, it should print the message *Multiple agents* as soon as possible. Describe your approach in the readme; the smarter your approach, the more credit you will achieve.
4. (10 points) Create a goal/subgoal named *circum-navigate* that enables your eater to walk around a wall. The goal (operator) should be proposed so you can avoid jumping over walls (thereby not loosing points for no good reason).
5. (10 points) Assume there is an eater which will eat 100 points and then sit at location (1,1) on the random map. Create a goal which will send your agent to attack that eater if it can gain points with this strategy. Be sure to clearly indicate in your readme how this decision is made.
6. (10 points) Tweak your eater so it PWNS the random agent (i.e., move-to-food.soar). Run the random map 10 times with both your eater and the random agent and put the average score/std dev into your readme. You can download the Soar version from the website (will be posted on friday) that allows each agent to produce a move each "turn".
7. (10 points) Backup your claim of superiority with some real data (you can use the same runs here as in the previous part). Tweak the random agent and your own agent so that they both print the eater's score each decision cycle (use the random feature on the input link to help out). Use excel (or whatnot) to plot the average score of your eater over 10 runs against the average score of the other eater on those same 10 runs. (setting watch 0 will help make the output clean)