CISC 3410 Fall 2010, Project 2

1 Description

This project works with the same world you used in Project 1, but develops some different mechanisms for finding the goal. Again you will develop code and carry out some experiments. In particular, you will:

1. Experiment with the program;
2. Extend the program to use more advanced techniques to decide how to move the robot; and
3. Evaluate how much more efficient the more advanced techniques are.

2 Get ready

1. Use the same file `robots.nlogo` from the course web site and open it in Netlogo.
2. Press **setup**.
3. The default is to set up a 25 by 25 patch world, with a randomly positioned robot (little robot symbol) and goal (red patch) , and 750 randomly position obstacles (black patches).
4. You will probably want to reduce the world to around 10 by 10.
5. You should also select **random** on the **movement** selector.
6. When you press **go** the robot makes random decisions about which square to move to next, and stops when it arrives at the goal.
7. Play with the simulation until you are comfortable with what it is doing.

3 Experiment

1. Use **setup** to generate a random location for the robot and the goal.
2. Press **go** and run the robot until it reaches the goal.
3. Make a note of the number of ticks it takes.
4. Repeat this 19 times, so you have a set of 20 measurements. Be careful to get the right number of ticks — reset does *not* reset the number of ticks so you will have to do a little math to get the right number.
5. Compute the average number of ticks and the standard deviation of the number of ticks.

   (If you don't know how to compute the standard deviation, look it up. The computation for a small sample like this is trivial in a spreadsheet).
6. In the **Information** tab, document the results from your experiment.

4 Extend

1. Look at the code in the **Procedures** tab.
2. The key functions are `robot-move` and `robot-random-move`. This is the place where the decision about how to move takes place.
3. You then need to modify the code to do the following:

(a) Use a heuristic to choose the next move rather than making a choice at random. You should have code for a suitable heuristic value from Project 1.

(b) Give the option of using movement using the heuristic as well as random movement to find the goal.
What I mean by "as well as" is that I want your program to give the option of using either the heuristic or random movement.

4. Then we get more sophisticated.

   (a) Add an implementation of the simple reinforcement from Lecture 9.

   (b) This method should run the learning procedure to compute a heuristic value for each patch, and should continue until the values are stable.

   (c) When the values are stable, the robot should use the value of each patch to find the goal.

   (d) Give the option of using reinforcement learning as well as the heuristic and random movement from before.
   What I mean by "as well as" is that I want your program to give the option of using all three methods.

5. While you are writing code, you may find it helpful to work with a much smaller world. However, when you are done you need to make sure that your code runs on a 10 by 10 patch world.

5 Evaluate

1. Evaluate your new search techniques by repeating the experiments you did for random movement for the use of the simple heuristic and reinforcement learning
   You need to do this on a 10 by 10 world so that your results are comparable with those for breadth-first search.

2. Again calculate the average number of visited nodes and the standard deviation.

3. For the reinforcement learning approach, just count the time it takes the robot to find the goal once the values have been learnt.

4. For additional credit, find a way to measure the time it takes to learn the values as well.

5. Add the results to the **Information** tab, and explain the differences (or lack of differences) between the three sets of results.

6 Finishing up

1. Make sure you document your code in the **Procedures** tab, and that you write up your experiments in the **Information** tab.

2. For additional credit, color the green patches for which a heuristic has been computed to reflect their heuristic value. This should show a color gradient leading to the goal.

3. Rename your program `robots2-<myname>.nlogo`, where `<myname>` is your name, so my program would be named `robots2-parsons.nlogo`.

4. Send your program to me.