CSE 512 – Winter 2018 – Independent Work

In place of lecture/lab on Jan 30, 2017

Instructor: Kerstin Voigt

In place of the lecture and lab, work independently on the following exercise.

Start by obtaining copies of files

```
bestfirst_astar_search.py
puzz8.p
```

Both files expand on our recent graphsearch implementation, and you should recognize a substantial portion of the code in file bestfirst_astar_search.py. An additional application domain, the "8-puzzle", is coded in file puzz8.py.

At the bottom of bestfirst_astart_search.py, you will find loops calling best-first search and a new variant called "A-star" search for one problem in the robot-in-rooms domain, and one problem (puzzE, see in puzz8.py) in the 8-puzzle domain. Notice how both search functions have function parameters that allow us to pass domain-specific "goal functions" (true when goal is reached), "successor functions" (computing next possible states), and "evaluation functions" (measuring estimated distance of state from goal) just like any other parameters.

```
print "\nBest-first search in robot domain:"
for i in range(5):
  BEST_FIRST_SEARCH_FCT('o103', 'r123', rob_goal, rob_next, rob_cost)

print "\nA* search in robot domain:"
for i in range(5):
  ASTAR_SEARCH_FCT('o103', 'r123', rob_goal, rob_next, rob_cost)


print "\nBest-first search in 8-puzzle domains:"
show_puzz8(puzzE)
for i in range(5):
  BEST_FIRST_SEARCH_FCT(puzzA, GOAL, puzz8_goal_fct, puzz8_successor_fct,\
                puzz8_eval_fct, puzz8_compact)

print "\nA* search in 8-puzzle domains:"
show_puzz8(puzzE)
for i in range(5):
  ASTAR_SEARCH_FCT(puzzA, GOAL, puzz8_goal_fct, puzz8_successor_fct,\
                puzz8_eval_fct, puzz8_compact)
```

You are invited to examine the program code and extract some sense and new Python knowledge from it. Yet, for this exercise, you may also skip directly to testing out the search functions. By virtue of loading/running bestfirst_astar_search.py, the above code is executed and the results are being listed in the interpreter window.

You will see the results of 5 rounds of best_first and Astar search for an example problem in the robot domain (here: finding goal 'r123' from starting state 'o103'). The more interesting runs follow when 5 rounds each are run for a given 8-puzzle problem (here: puzzA). Just load, run and see what you get.

Then edit the file by replacing 'puzzA' with 'puzzB'; load and observe the reported results.

Repeat with puzzle problems puzzC, puzzD, and puzzE. Observe and notice the performance data that are being listed for puzzles of increasingly greater degree difficulty.

One particular feature of Astar search is that, similar to breadth-first search, it *is guaranteed to produce the solution that lies on the shortest path to the starting state.* When the path length for the solution to problem puzzE is reported (under PathL: …), verify by hand, that the puzzle cannot be solved in fewer steps (no formal proof, just do your cleverest best and see whether you can beat the reported number of solution steps for this puzzle).

The bodies of search functions BEST_FIRST_SEARCH_FCT and ASTAR_SEARCH_FCT look. Yet they produce different output (at least for some puzzles), and *there must be a difference.* Spend some time examining both functions, and see if you find the portions of the code which accounts for the difference. What you should find is subtle but important for the special properties of the Astar search functions.

On Thursday, we will try to understand in greater depth what is going on. Until then, experience the search functions life, take down some performance measures, do some thinking, and come prepared to discuss and share your observations in the lecture on Thursday.

There is nothing to be handed in, but be assured that spending time with this exercise will (a) be fun and (b) propel you along with your learning of some fundamental AI algorithms. You can use our lecture and lab time in JB 359 to do this.