CS F407 Artificial Intelligence Assignments

SUJITH THOMAS   Ramprasad S Joshi

August 30, 2019

Abstract

The lab credits are to be divided up as follows:

1. A programming exercise on the Search Module (Chapters 2 through 6). **[10 Marks]**
2. A programming exercise on a subsequent module (TBA). **[10 Marks]**

The mode of the assigment – individual or group, etc. – and of evaluation will be declared for each independently.

Contents

1 Search Module

The first assignment of 10 marks is

to modify the A* algorithm with a dose of local (limited depth) BFS.

The idea: In the A* algorithm, when a node is expanded, expand up to a limited (fixed, constant) depth by the BFS strategy. After expansion, add the nodes not in the BFS (FIFO) order but in the usual A* order.

1.1 The Task

Divide the implementation in 5 parts (so that you can get partial marks for each part completion):

1. Implement the usual A* for 8-PUZZLE, with the EXPAND function taking a "`depth`" parameter – the depth to which each expand call expands.
2. Implement both the usual heuristics: tile mismatch count and manhattan distance.[1]
3. In the `main` function, read the `depth` parameter as input, get the INITIALSTATE, and call the A* solver **twice** for each heuristic: once with the input `depth` parameter, and then again with the parameter being 1 (which is the usual EXPAND).
   Thus the total number of times that the puzzle will be solved by A* is 4.
4. Compare the results of all the four calls for at least 10 randomly generated initial states for optimality. Make sure this is done in the program itself.
5. Compare the computational costs (number of nodes generated and the maximum length of the fringe throughout) of all the four calls for the different initial states.

[1]Apart from the INITIALSTATE as the usual parameter and the `depth` parameter being an extra input to A*, also take the choice of the heuristic function as one more parameter.

1.2 Mode

Individual submissions, either on moodle link or mail; that part will be declared at opportune time, with the deadline. Copying will be summarily punished and no time will be spent on arguments and negotiations on that.

1.3 Evaluation

As suggested above, partial marks for each part completion. It must run without hassle on a GNU/Linux platform. No barrier on the choice of a programming language, but if we cannot compile and execute your program without buying licenses, then porting to our platform is not our responsibility. The burden of proof of completion is yours. Supply one zipped folder that must expand to a unique name, we should be able to identify you at a glance at the folder, and supply one `makefile` or a shell script in it such that our issuing `make` or the shell script name should be sufficient for our evaluation process.

1.4 The Skeleton Algorithm

Algorithm 1 describes the A* modification, whereas Algorithm 2 describes the EXPAND modification.

Algorithm 1 ModifiedA*

```
Require: $I$: The Intial State
Require: $h$: The Heuristic Function
Require: depth: The BFS Depth
 1: fringe $\leftarrow$ []
 2: $S \leftarrow I$
 3: $f \leftarrow 0$
 4: while $\neg$GOAL-TEST($S$) do
 5:     temp $\leftarrow$ MODIFIEDEXPAND($S$, depth)
 6:     for each $T \in$ temp do
 7:         UPDATEFRINGE(fringe, $T$, $f = h(T) + g(T)$)
 8:     end for
 9:     $S \leftarrow$ REMOVEFIRST(fringe)
10: end while
11: return $S$
```

Algorithm 2 ModifiedExpand

```
Require: $S$: The State to be Expanded
Require: depth: The BFS Depth
 1: $\mathtt{temp}_1 \leftarrow$ MAKEFIFOQUEUE(EXPAND($S$))
 2: $\mathtt{temp}_2 \leftarrow \mathtt{temp}_1$
 3: $d \leftarrow \mathtt{depth} - 1$
 4: while $d > 0$ do
 5:     while $\neg$EMPTY($\mathtt{temp}_2$) do
 6:         $C \leftarrow$ REMOVEFIRST($\mathtt{temp}_2$)
 7:         $\mathtt{temp}_3 \leftarrow$ EXPAND($C$)
 8:         $\mathtt{temp}_1 \leftarrow$ APPEND($\mathtt{temp}_1$, $\mathtt{temp}_3$)
 9:     end while
10:     $d \leftarrow d - 1$
11:     $\mathtt{temp}_2 \leftarrow \mathtt{temp}_3$
12: end while
13: return $\mathtt{temp}_1$
```