Haley Williams
hw481
CSE 4633
Oct 17, 2018

Anytime Weighted A*

A & Weighted A**

The A* algorithm begins by utilizing two lists, an Open list and a Closed list. These lists are used to manage a "systematic search for a minimum-cost path from a start node to a goal node in a graph". At the start of the algorithm, the Open list contains the start node, and the Closed list is empty. For each iteration of the cycle, the most promising node is expanded, moved to Closed list, and successor nodes are put into the Open list. The Closed list contains the expanded nodes, and the Open list contains the nodes to be expanded. The search ends when the goal node is added to the Closed list. A solution path is found by tracing node pointers backwards from goal node to start node. The node evaluation function is $f(n) = g(n) + h(n)$.

Weighted A* (WA*) is different from A* in that the terms $g(n)$ and $h(n)$ are weighted. The node evaluation function is $f'(n) = g(n) + w \times h(n)$ where $w >= 1.0$ is a parameter set by the user. WA* is faster than A* because during the solution search, it makes the nodes that are closer to a goal seem more attractive. The algorithm gives the search a depth-first search and has a tradeoff between search effort and solution quality.

*Anytime Weighted A**

Anytime Weighted A* (AWA*) is a algorithm created by Hansen and Zhou. The AWA* algorithm is much like the Weighted A* (WA*) with three major changes, listed below:

1) *Non-admissible function f'(n) = g(n) + h'(n), where h'(n) is not admissible, is used to select nodes for expansion in an order that allows good, possibly suboptimal, solutions to be found quickly*
2) *Search continues after solution is found, in order to find improved solutions*
3) *Admissible function f(n) = g(n) + h(n), where h(n) is admissible, is used together with upper bound on optimal solution cost in order to pounce search space and detect convergence to optimal solution*

A weighted heuristic is used to create the non-admissible evaluation function that leads the search. An admissible heuristic h(n) is assumed and used to create a weighted heuristic $h'(n) = w \times h(n)$. The three steps that are listed above can be used to find any non-admissible heuristic that helps A* find a solution more quickly - that is, it is not limited to a weighted a heuristic search. When the general approach is used to alter A* into an anytime algorithm, the resultant algorithm is known as Anytime A*. The algorithm only becomes known as AWA* when the non-admissible evaluation function is a weighted heuristic.

AWA* continues the search after the first solution is found and also prunes the search space. As improved solutions are found by the algorithm, the upper bounds are improved as well. The algorithm tests whether the f-cost of each newly-generated node is less than the current upper

bound; if it is not less, than the node is inserted into the Open lost, therefore reducing the memory requirements of the algorithm.

There are two reasons why AWA* needs more node expansions than A*. The first reason why Is that the use of a weighted heuristic allows the algorithm to expand more distinct nodes than A(. The second reason why is that a weighted heuristic is inconsistent, which means that it is possible for a node to have a higher-than-optimal g-cost when expanded. If a better path is found to the same node later on, then the node is reinserted back into the Open list, and the improved g-cost is able to be propagated to its descendants when the node is reexpanded. AWA* can expand the same node multiple times.

Changes Made to Code

The first thing I changed was adding a node to a closed list if the incumbent is equal to none or if the node's amount is less than the incumbent's amount. The next thing I changed was that I added a weighted heuristic that could be calculated from the children (child.wh = child.h * 1). I then looped through the children node list and added the valid children to an Open list. This is done when incumbent is none or the child's amount is less than the incumbent's amount. If the child is the goal node, the incumbent is then set to the goal node. If the child is both in the closed list and the child.c is less than the child in the closed list, the code then adds that child to the Open list and removes it from the Closed list. Else, that child is added to the Open list. The output returns the incumbent, the length of the Closed list, and the length of the Open list.

Puzzle to Be Solved

The puzzle that my code uses is the sliding tile puzzle. It is contained in the puzzle.py file. It builds the puzzle using a class and contains how to move the puzzle around. You are able to change the state, width, and heuristic from the command line.

Comparison between A and AWA**

The output from left to right: Size of sliding tile puzzle, the path that was taken, how many moves it took to get to a solution, the heuristic that was used, if IDA* flag is triggered, how many nodes are expanded and on the Closed list, how many nodes that are remaining and on the Open list, and the run time in seconds.

A* Output, no weighted heuristic, sliding tile = 8

```
8, 2 6 7 3 5 1 4 _ 8, 23, manhattan, False, 910, 519, 0.0878758430480957
8, 6 5 8 1 3 2 _ 7 4, 26, manhattan, False, 1893, 1031, 0.10048198699951172
8, 4 1 2 3 _ 6 5 7 8, 12, manhattan, False, 17, 14, 0.0007679462432861328
8, 6 2 7 8 5 _ 4 3 1, 25, manhattan, False, 494, 273, 0.029426097869873047
8, 3 7 5 4 1 2 6 8 _, 24, manhattan, False, 1065, 603, 0.0837099552154541
```

AWA* Output, weighted heuristic = 1, sliding tile = 8

```
8, 2 4 7 6 5 _ 3 8 1, 29, manhattan, False, 3273, 0, 0.772212028503418
8, 2 6 5 4 _ 1 3 7 8, 20, manhattan, False, 466, 0, 0.2774851322174072
8, _ 7 1 8 4 6 3 5 2, 24, manhattan, False, 564, 0, 0.1263258457183838
8, 5 4 1 3 7 _ 2 6 8, 25, manhattan, False, 1345, 0, 0.4976990222930908
8, 4 7 5 1 2 6 3 _ 8, 23, manhattan, False, 540, 0, 0.1538538932800293
```

For the sliding 8 tile puzzle, A* ran faster and took less moves than AWA*. The reason for this is that the AWA* program explored all options and left no nodes closed; the algorithm wanted to make sure that it did find the optimal solution. The A* algorithm, however, only explored the child nodes that looked promising. AWA*, although it takes more time, tends to find the more optimal solutions over A*.

References

Anytime Heuristic Search
https://arxiv.org/pdf/1110.2737.pdf

Zach A*
https://github.com/ZDTaylor/Class-Project-Astar