CSE537 Artificial Intelligence

Assignment-1 Report

Peg Solitaire using IDS and A*

Author: Vishal Nayak

SBU-ID: 109892702

Email:

vishal.nayak@stonybrook.edu

vnayak@cs.stonybrook.edu

Contents

1. IDS

This algorithm uses the depth first search algorithms with a modification that it runs iteratively covering different depth levels of the search at each iteration of the algorithm. The code uses python list as an implementation of stack and uses it as fringe to put nodes into it and remove it from the front every time it is expanding the node. This was not efficient. If the problem size is big, then the algorithm fails to reach the goal state. I saw cases like recursion limit getting reached, program crashing and in some cases segmentation faults. The number of nodes expanded is very huge.

2. A*

A* uses priority queue instead of stack for picking the node from the fringe. In the code, I am using *heapq* implementation which takes a supplied python list and uses it to maintain the elements in the form of a min heap. I am using that as a priority queue fringe.

The cost function of the priority queue (*f(n)*) for this algorithm takes in two items: *g(n)* and *h(n)*. *g(n)* is just the distance of the node from the start state. *h(n)* is the estimate of the current node's distance from the goal. Two heuristics used in *A** are *h1(n)* and *h2(n)*. Both are admissible and also *h2(n) > h1(n)*.

2.1. A* with Heuristic-1

A node whose state has highest possible options to go for is chosen first. In other words, a state with maximum possible successors is chosen with higher priority. This is because, more the number of states more moves are possible. More moves means that more pegs will be removed from the state, nearing to the goal faster. This is an admissible heuristic because all the heuristic values of all the nodes cannot exceed an upper limit. i.e,. 31

2.1. A* with Heuristic-2

A state with less sparse pegs is chosen with higher priority. A sparse peg is the one which has no moves, either because it is in the border of the board and/or it has no neighbors. If there are sparser pegs on the board, chances are likely that they will not be moved, unless other pegs reach the sparse peg's neighboring position, enabling them to move. Nevertheless, the number of sparse pegs on the board serves as a fine heuristic for A* search. This is admissible as well since the number of sparse pegs cannot exceed the number of pegs the board can have. Also, for the pegs to be sparse, they should not have neighbors. This essentially means that the upper limit for this heuristic should be less than the upper limit of the previous heuristic.

3. Solved Test Cases

3.1. Test Case 1

XX000XX
XX010XX
0000010
0010110
0000100

XX000XX
XX000XX

3.2. Test Case 2

XX000XX
XX010XX
0011100
0001000
0001000
XX000XX
XX000XX

3.3. Test Case 3

XX000XX
XX010XX
0000100
0011110
0001100
XX000XX
XX000XX

3.4. Test Case 4

XX000XX
XX010XX
0100100
0111110
0010100
XX000XX
XX000XX

4. Statistics

Observations:

1) The number of nodes expanded by IDS method are very huge.
2) The number of nodes expanded by A* method is relatively less when compared to IDS.
   Note: A* worked well because of the admissible heuristics chosen.
3) A* with heuristic-1 reduced the number of nodes expanded (relative to IDS) drastically.
4) A* with heuristic-2 reduced the number of nodes expanded (relative to A* with heuristic-1) even more.
5) In relation to heuristic-1 and heuristic-2, the differences in the number of nodes expanded were not much when the goal state is closer to the start state. But as the problem complexity increased, A* with heuristic-2 performed much better.

Below is the pictorial representation of the trace log data.