CS26010 - Assignment 2010/11

Alexander David Brown

November 16, 2010

1 Assignment Part 1

a) **X1:** 3

X2: 5

X3: 2

Y1: 5

b) **Best move for MAX:** A2

Best move for MIN: A21

c) **Unexamined Branches:** A32

Explanation: All nodes at *X1* would be opened, the minimum of these is 3. All of *X2*s branches would then be opened as none of them are less than 3. Finally, at *X3*, *A31* and *A32* would be opened. As the value at *A32* is 2. As 2<3, the final branch, *A33* is not opened.

d) Whilst uninformed search is good in a constrained environment, the algorithms it uses typically have an exponential time complexity. When a problem reaches a certain size it is not feasible to use brute force.

Heuristics can give us a workable solution in a reasonable amount of time, allowing for a more practical way of determining a solution for which a brute force algorithm might spend several hours finding the best solution.

Heuristics (like the Hill Climbing Algorithm) can also allow us to deal with infinite search states as not all states are explored as the heuristic function guides the search.

Heuristics are also useful for graph based searches where expanding every node may not be feasible, or for problems like the Traveling Salesperson where the number of combinations and permeations is factorial.

e) Heuristics like the hill climbing algorithm tend to return to a local minima, rather than the global minima. This tends to leave the algorithm stuck in a sub-optimal solution.

This can be solved using restarting to a random location and running the algorithm several times, saving the lowest found values. Again this might not find the global minima, but it is more likely to find a local minima closer to the global and in a reasonable amount of time.

Plateaus can also skew the results of the hill climbing algorithm, as there is no one 'right' way in which to move. Most algorithms use a random walk to cope with this.

One can also apply simulated annealing to the hill climbing algorithm. When a local minima is found the algorithm then 'jumps' to another point on the state space and starts again. This then repeats with consecutively smaller jumps and typically a better result is found.

Heuristics are only as good as the heuristic function which they use. However, a heuristic function might give a perfect result but it could also take more time to produce said result that it takes longer than an uninformed search.

It should be noted that the weaknesses of heuristics are often strengths as well.

2 Assignment Part 2

2.1 Notes

2.1.1 Identifiers

b The number of moves that can be made ($p-1$ in most cases).

d The depth of the solution ($(p-1)^n-1$).

l The depth limit set for the Depth Limited Search. (d to get the best solution).

m The maximum depth of the solution (potentially ∞ due to the way a search tree would loop the solution).

n The number of disks.

p The number of poles (3 typically).

2.1.2 Non-applicable Algorithms

The Uniform Cost Algorithm is not applicable to this problem as the path cost is always one and will therefore act as a Breadth First Search.

2.2 Breadth First Search

2.2.1 Time Complexity

The time complexity for the Breadth First Search is b^d.

2.2.2 State Space

The state space for the Breadth First Search is b^d.

2.2.3 Completeness

The Breadth First Search will give a complete solution to the Towers on Hanoi.

2.2.4 Optimality

The Breadth First Search will produce an optimal solution for the puzzle.

2.3 Depth First Search

2.3.1 Time Complexity

The time complexity for the Depth First Search is b^m.

2.3.2 State Space

The state space for the Depth First Search is bm.

2.3.3 Completeness

The Depth First Search will not give a complete solution to the Towers on Hanoi and will likely get trapped in loops.

2.3.4 Optimality

The Depth First Search will not produce an optimal solution for the puzzle.

2.4 Depth Limited Search

2.4.1 Time Complexity

The time complexity for the Depth Limited Search is b^l.

2.4.2 State Space

The state space for the Depth Limited Search is bl.

2.4.3 Completeness

The Depth Limited Search will give a complete solution to the puzzle, assuming $l \geq d$.

2.4.4 Optimality

The Depth First Search will not produce an optimal solution for the puzzle.

2.5 Iterative Deepening

2.5.1 Time Complexity

The time complexity for Iterative Deepening is b^d.

2.5.2 State Space

The state space for Iterative Deepening is bd.

2.5.3 Completeness

Iterative Deepening will give a complete solution to the puzzle.

2.5.4 Optimality

Iterative Deepening will produce an optimal solution for the puzzle.

2.6 Bi-directional Search

2.6.1 Time Complexity

The time complexity for the Bi-directional Search is $b^{d/2}$.

2.6.2 State Space

The state space for the Bi-directional Search is $b^{d/2}$.

2.6.3 Completeness

The Bi-directional will give a complete solution to the puzzle.

2.6.4 Optimality

The Bi-directional will produce an optimal solution for the puzzle.

2.7 Comparison

For $p = 3$ and $n = 3$

2.7.1 Breadth First Search

Time Complexity $= 2^{(2^3-1)} = 2^7 = 128$
Space Complexity $= 2^{(2^3-1)} = 2^7 = 128$
The Breadth First Search is one of the simplest searches to run on the Towers of Hanoi puzzle as it can be implemented with ease and will always give a optimal and complete result. However it has the drawbacks of having a large time and space complexity. Even for a small number of disks it is fairly inefficient, especially when contrasted to other search algorithms.

2.7.2 Depth First Search

Time Complexity = $2^{(2^{\infty}-1)} = 2^7 = \infty$
Space Complexity = $2 \times \infty = \infty$
Due to the nature of the Towers of Hanoi puzzle, a Depth First Search is not a useful search as the puzzle tends to loop a lot. This makes the time and space complexity of the Depth First Search potentially limited as it is prone to getting stuck in loops.

2.7.3 Depth Limited Search

Time Complexity = $2^{(2^3-1)} = 2^7 = 128$
Space Complexity = $2 \times (2^3 - 1) = 2 \times 7 = 14$
The problems with the Depth First Search can be countered by using a Depth Limited Search. This is one of the more favorable search techniques for this puzzle as the depth of the solution is always known. The time complexity is still exponential and therefore still inefficient for a larger number of disks.

2.7.4 Iterative Deepening

Time Complexity = $2^{(2^3-1)} = 2^7 = 128$
Space Complexity = $2 \times (2^3 - 1) = 2 \times 7 = 14$
Iterative Deepening is another way to rectify the problems of the Depth First Search, however it is not as useful as a Depth Limited Search as the depth of the puzzle is actually know. It should be noted that due to the iterations involved with Iterative Deepening, it is actually slower that the Depth Limited Search, though it is still exponential.

2.7.5 Bi-directional Search

Time Complexity = $2^{(2^3-1)/2} = 2^{7/2} = 12$
Space Complexity = $2^{(2^3-1)/2} = 2^{7/2} = 12$
Bi-directional Search is probably the most relevant of the uniformed searches, as both the start and goal state are well defined. This is reflected by a relatively low time complexity. For a larger number of disks it may be best to chose the Depth Limited Search for it's good space complexity.

2.8 A word on heuristic and other algorithms

I haven't included heuristics in the comparisons as most goal functions would generally make it more difficult to find the solution. Also as the monks are looking for a complete solution heuristics have limited use as they are better at finding a decent solution to a problem rather than a complete one.
It should also be noted that working out an algorithm for this problem would actually be the best route as the puzzle lends itself those solutions.