
   Key Ideas:
   - “Effect” of evolution can be reduced to an optimization problem
   - Evolution is linked to our concept of rationality

   Evolution tends to perpetuate organisms (and combinations and mutations of organisms) that are successful enough to reproduce. That is, evolution favors organisms that can optimize their performance measure to at least survive to the age of sexual maturity and then be able to win a mate. Rationality means optimizing some performance measure, so this is in line with evolution.

2. General AI: Exercise 1.11 of Russell & Norvig.

   Key Ideas:
   - “Intelligence” can mean many different things
   - How you define Intelligence is key

   The statement that computers can do only what their programmers tell them is true in a sense since computers perform our set of instructions and they cannot think for themselves or pass the Turing test. However, depending on how you define intelligence, this doesn’t necessarily have to imply that computers cannot be intelligent. Computers may not have common sense or the same kind of intelligence that animals process but if you define intelligence as the ability to acquire and apply knowledge and skills then computers can learn from what they do and they can do things computationally that humans cannot do on their own. If you define intelligence more like what makes animals intelligent then computers do not possess this. Any reasonable answer should be fine for this question.
3. Water Jug Puzzle:

(a) Formulation:

Key Ideas:
• Complete description of all possible stages
• Detailed operator definitions (no room for ambiguity)
• Start and goal states identified
• Reasonable path cost

Example:
• States: \((x,y)\) where \(x \in \{0,1,2,3\}\) and \(y \in \{0,1,2,3,4\}\). \(x\) would represent liquid in the jug with capacity 3 and \(y\) would represent liquid in the jug with capacity 4. This creates a situation with 20 possible configurations where 14 of them are reachable.
• Set of reachable states: \(S_0 = (0,0), S_1 = (0,4), S_2 = (3,0), S_3 = (0,4), S_4 = (0,3), S_5 = (3,1), S_6 = (1,0), S_7 = (1,4), S_8 = (3,2), S_9 = (0,2), S_{10} = (2,0), S_{11} = (0,4), S_{12} = (3,3), S_{13} = (0,1)\)
• Operators (6 of them):
  – A: Empty THREE (drain)
  – B: Empty FOUR (drain)
  – C: Fill THREE (tap)
  – D: Fill FOUR (tap)
  – E: Fill FOUR with THREE (not allowing any water to spill)
  – F: Fill THREE with FOUR (not allowing any water to spill)
• Start State: \((0, 0)\)
• Goal State: \((\sim, 2)\), only \((0, 2)\) and \((3, 2)\) are reachable.
• Path Cost: Cost of 1 for all iterations. Generally, any constant cost for all operations should work. However, cost of 0 does not work because it could lead to an infinite loop.
Key Ideas:
• All edges should be labeled with operators
• Path to goal states should be provided
• Directed graph with logical transitions
• No duplicate nodes (instead use backward edges)
• Note: More detail given here than necessary
4. Uninformed Search:

(a) The fundamental difference between tree search and graph search is that graph search keeps track of the nodes that have been visited (explored set). This means that it is guaranteed to terminate when run on a generally finite graph. Tree search would not necessarily terminate because it could follow a cycle indefinitely.

(b) The if-else statement in the uniform cost search algorithm simply guarantees that if a node in the frontier can be reached from more than one explored node, the algorithm will explore this node by following the edge of minimum cost. Graph search tends to just look for the shortest path that does not account for the actual cost of each edge. In the example below, we start at A. B is added to the frontier with cost 200 and C is added with cost 50. C is explored first because its edge has lower cost. Once C has been explored, we see that B can be reached with a cost of 100, so its cost in the frontier is changed from 200 to 100.
(c) The given statement is **false**. Fundamentally, uniform cost search works because shortest cost paths can be composed while longest paths cannot. The main issue is that uniform cost search follows a local strategy but a global strategy is required to find the longest path. (This ties into the idea that uniform cost search requires that all edge costs are positive) In uniform cost search, a goal node is only explored if every other frontier node has higher cost, so a different path with lower cost to the goal node will never be found. If we explore the highest cost node and find a goal node, we have no guarantee that lower cost frontier nodes do not lead to a goal node on a path with higher cost.

We can use the general structure of part (b) as a counter-example. Simply replace the 50 cost edges with 150 cost edges. Starting with A, the algorithm will add B to the frontier with cost 200 and C to the frontier with cost 150. Since B has higher cost, it will be explored first by following the edge with cost 200. Since B is the goal node, the algorithm will terminate once it is reached. Thus, the algorithm will find the path AB which has cost 200. However, the path AC B has path cost 300 and it is not found, so we can conclude that this algorithm does not find the longer path from the start to the goal.
5. Heuristic Search:

(a) Uniform-Cost Search:

Expanded Nodes: SBACDEFGZ
Path to Goal: SADGZ
Frontier at each time step:

1  \{ (S, 0) \}
2  \{ (B, 2), (A, 3), (C, 5) \}
3  \{ (A, 3), (C, 5) \}
4  \{ (C, 5), (D, 6), (E, 7) \}
5  \{ (D, 6), (E, 7), (F, 8) \}
6  \{ (E, 7), (F, 8), (G, 10) \}
7  \{ (F, 8), (G, 10) \}
8  \{ (G, 10) \}
9  \{ (Z, 11) \}

(b) Greedy Search:

Expanded Nodes: SCFEGZ
Path to Goal: SCEGZ

(c) A* Search:

Expanded Nodes: SBACDEFGZ
Path to Goal: SADGZ
Frontier at each time step:

1  \{ (S, 8) \}
2  \{ (B, 8), (A, 10), (C, 10) \}
3  \{ (A, 10), (C, 10) \}
4  \{ (C, 10), (D, 10), (E, 10) \}
5  \{ (D, 10), (E, 10), (F, 10) \}
6  \{ (E, 10), (F, 10), (G, 11) \}
7  \{ (F, 10), (G, 11) \}
8  \{ (G, 11) \}
9  \{ (Z, 11) \}
6. Game Tree Search: *Note: We could also accept ACFLY, ACGODD or ACGOEE as a winning path for the Maximizer Player in the case that the minimizing player decides to not play rationally (but they should probably just assume that the minimizer plays rationally).*

(a) Next Move: C, Expected Value: 3. This assumes path ACFLX

(b) 10 leaves evaluated. Optimal path is ACFLX.

(c) 11 leaves evaluated. Optimal path is ACFLX

(d) The number of nodes that are skipped could be different if a different evaluation order is used. For instance, if we use perfect ordering (examine first the best successors), only 7 leaf nodes are evaluated. Generally, if we assume the depth of the tree is \(d\) and that there are \(b\) legal moves at each point then the alpha beta pruning with perfect ordering only examines \(O(b^{d/2})\) nodes to pick the best move, instead of \(O(b^d)\) with standard minimax.

(e) There may be many possible goal states so backward-search would have to either choose (with the possibility of choosing wrong) or try them all. Many of these goals may be unattainable, so searching from them is a waste of time. Finally, in many cases, the distance from the goal node is so far that there would not be enough resources to search back to the current state.
7. Game Tree Search:

(a) Move (3) will be chosen since it has the highest expected value: 0.1/6

(b) Move (3) is the only option that has a 1/6 chance of resulting in an immediate loss. There are many acceptable ways to deal with this. Here are a few options:

- Take standard deviation into account and try to maximize the expected value and minimize the standard deviation at the same time. Specifically, suppose that X is a random variable representing the score based on the die. We can choose a move that maximizes $E[X] - C(\text{Var}(X))^{1/2}$ where $C$ is some given constant.

- Choose a movie that maximizes expected score but does not potentially lead to a loss.

- Incorporate a higher penalty than -1 for moves that lead to a loss.