Artificial Intelligence: Week 2

1 Problem Solving by Searching

Goal formulation, based on the current situation and the agent's performance measure, is the first step in problem solving. **Problem formulation** is the process of deciding what actions and states to consider, given a goal. In general, *an agent with several immediate options of unknown value can decide what to do by first examining future actions that eventually lead to states of known value.* The process of looking for a sequence of actions that reaches the goal is called **search**. A search algorithm takes a problem as input and returns a **solution** in the form of an action sequence. Once a solution is found, the actions it recommends can be carried out. This is called the **execution** phase. Note while the agent is executing, it *ignores its percepts* when choosing its actions because it knows in advance what they will be.

Together, the **initial state, actions and transition model** implicitly define the **state space** of the problem - the set of all states reachable from the initial state by any sequence of actions. A **solution** to a problem is an action sequence that leads from the initial state to a goal state. Solution quality is measured by the path cost function, and an **optimal solution** has the lowest path cost among all solutions. The process of removing detail from a representation is called **abstraction**.

It is quite important to distinguish between a node and a state: a node is a bookkeeping data structure used to represent the search tree. A state corresponds to a configuration of the world. Furthermore, two different states can contain the same world state if that state is generated via two different search paths. We can evaluate the performance of an algorithm in four ways:

- **Completeness**: Is the algorithms guaranteed to find a solution if there is one?
- **Optimality**: Does the strategy find the optimal solution?
- **Time complexity**: How long does it take to find a solution?
- **Space complexity**: How much memory is needed to perform the search?

In AI, the graph is often represented implicitly by the initial state, actions and transition model and is frequently infinite. Complexity is expressed in terms of

three quantities: b, the **branching factor** or maximum number of successors of any node; d, the **depth** of the shallowest goal node; and m, the maximum length of any path in the state space.

1.1 Uninformed Search Strategies

1.1.1 Breadth-First Search

This is a simple strategy in which all the nodes are expanded at a given depth in the search tree before any nodes at the next level are expanded. The algorithm is given below:

```
function BFS(problem) returns a solution, or failure
  node = a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
  if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
  frontier = a FIFO queue with node as the only element
  explored = an empty set
  loop do
    if EMPTY?(frontier) then return failure
    node = POP(frontier)
    add node.STATE to explored
    for each action in problem.ACTIONS(node.STATE) do
      child = CHILD-NODE(problem, node, action)
      if child.STATE is not in explored or frontier then
        if problem.GOAL-TEST(child.STATE) then return SOLUTION(child)
          frontier = INSERT(child, frontier)
```

BFS is *complete* - if the shallowest goal node is at some finite depth BFS will eventually find it after generating all the shallower nodes. Note that the *shallowest* is not always the *optimal* one. BFS is only optimal if the path cost is a non-decreasing function of the depth of the node. The time complexity of BFS is:

$$b + b^2 + b^3 + ... + b^d = O(b^d) \tag{1}$$

For the space complexity there will be $O(b^{d-1})$ nodes in the explored set and $O(b^d)$ nodes in the frontier, giving space complexity as $O(b^d)$. *Memory requirements are a bigger problem for BFS than execution time and time is still a major factor. Exponential-complexity search problems cannot be solved by uninformed methods for any but the smallest instances.*