AI

What is Intelligence?

– The capacity to acquire and apply knowledge.
– The faculty of thought and reason.

Artificial Intelligence:

-Studies how to achieve intelligent behavior through computational means

The Turing Test:

– A human interrogator. Communicates with a hidden
subject that is either a computer system or a human.
If the human interrogator cannot reliably decide whether or not the subject is a computer, the computer is said to have passed the Turing test.

In general, there are various reasons why trying to mimic humans might not be the best approach to AI:

– Computers and Humans have a very different architecture with quite different abilities.
– Numerical computations
– Visual and sensory processing
– Massively and slow parallel vs. fast serial
Neuroscience has been very influential in some areas of AI. For example, in robotic sensing, vision processing

Humans might not be best comparison?

– Don't always make the best decisions
– Computer intelligence can aid in our decision making

Rationality: Typically, this is a precise formal notion of what it means to do the right thing in any particular circumstance. Provides:
– A precise mechanism for analyzing and understanding the properties of this ideal behavior we are trying to achieve.
– A precise benchmark against which we can measure the behavior the systems we build.

AI tries to understand and model intelligence as a computational process.

Thus, we try to construct systems whose computation achieves or approximates the desired notion of rationality

Subareas of AI:

-Perception: vision, speech understanding, etc.
-Machine Learning, Neural networks
-Robotics
-Natural language processing
-Reasoning and decision making
-Many of the popular recent applications of AI in industry have been based on Machine Learning, e.g., voice recognition systems on your cell phone.
-Probabilistic graphical models are fundamental in machine learning.

AI Success:

- **Games**: chess, checkers, poker, bridge, backgammon... – Search
- **Physical skills:** driving a car, flying a plane or helicopter, vacuuming...

– Sensing, machine learning, planning, search, probabilistic reasoning

- **Language:** machine translation, speech recognition, character recognition,..

– Knowledge representation, machine learning, probabilistic reasoning

- **Vision:** face recognition, face detection, digital photographic
- **Commerce and industry:** page rank for searching, fraud detection, trading on financial markets...

– Search, machine learning, probabilistic reasoning

Formalisms and algorithmic ideas have been identified as being useful in the construction of these “intelligent” systems.

Search:

1-One of the most basic techniques in AI
2-Can solve many problems that humans are not good at (achieving super-human performance)
3-Very useful as a general algorithmic technique for solving many non-AI problems.
Search is a computational method for capturing a particular version of this kind of reasoning.

why search:

1- Successful:

- Success in game playing programs based on search.
- Many other AI problems can be successfully solved by search.

2- Practical:

- Many problems don't have specific algorithms for solving them. Casting as search problems is often the easiest way of solving them.
- Search can also be useful in approximation (e.g., local search in optimization problems).

Some critical aspects of intelligent behavior, e.g., planning, can be naturally cast as search.

Limitations of Search:

- Search only shows how to solve the problem once we have it correctly formulated.

formulate a problem as a search problem:

1-state space
2-actions
3-initial state and goal
4-heuristics

Inputs for search algorithms:

1-initial state
2-successors
3-goal test
4-actions cost

Outputs:

-a sequence of states leading from the initial state to a state satisfying the goal test.

Template Search Algorithms:

-The search space consists of **states** and actions that move between states.
-A **path** in the search space is a **sequence** of states connected by actions.
-The search algorithms perform search by examining alternate paths of the search space. The objects used in the algorithm are called nodes—each node contains a path.

We maintain a set of Frontier nodes also called the OPEN set.
These nodes are paths in the search space that all start at the initial state.

Selection Rule:

The order paths are selected from OPEN has a critical effect on the operation of the search:

-Whether or not a solution is found
-The cost of the solution found.
-The time and space required by the search.

All search techniques keep OPEN as an ordered set (e.g., a priority queue).

Critical properties of Search:

1-Completeness (will it find the solution if it exists)
2-Optimality (Will it always find the least cost solution)
3-Time complexity (what is the max number of paths that can be generated)
4-Space Complexity (what is the max number of paths that have to be stored)

Uninformed search techniques:

1-Breadth-First. 2-Depth-First. 3-Uniform-Cost 4-Depth-Limited
5-Irerative-Deepening search

Breadth-First: Place the new paths that extend the current path at the end of OPEN.

Completeness:

1- The length of the path removed from OPEN is non-decreasing
2- **All** shorter paths are expanded prior before any longer path.
3- Hence, eventually we must examine all paths of length d, and thus find a solution if one exists

Optimality:

1- will find shortest length solution
2- shortest solution not always cheapest solution if actions have varying costs.

Time Complexity: O(b^d+1)
Space Complexity: O(b^d+1)
Space complexity is a real problem.
Typically run out of space before we run out of time in most applications.

Uniform-Cost Search: Keep OPEN ordered by increasing cost of the path.

• Identical to Breadth first if each action has the same cost.

Completeness:

1-If each transition has costs $\geq \varepsilon > 0$.

Optimality:

1-Finds optimal solution if each transition has cost $\geq \varepsilon > 0$.

2- Explores paths in the search space in increasing order of cost.

3- So must find minimum cost path to a goal before finding any higher costs paths.

Time Complexity: $O(b^{\wedge}(C^*/\varepsilon))$ where C^* is the cost of the optimal solution.

Space Complexity: $O(b^{\wedge}(C^*/\varepsilon))$

when a path to a goal state is expanded the path must be optimal (lowest cost).

There may be many paths with cost $\leq C^*$: there can be as many as bd paths of length d in the worst case.

Depth-First Search: Place the new paths that extend the current path at the front of OPEN.

Completeness:

1- Infinite paths? Cause incompleteness!

2- Prune paths with cycles (duplicate states) We get completeness if state space is finite

Optimality: NO!

Time Complexity: $O(b^{\wedge}m)$ where m is the length of the longest path

Very bad if m is much larger than d (shortest path to a goal state)

Space Complexity: O(bm), linear space

A significant advantage of DFS

Depth-Limited Search:

- Breadth first has space problems, Depth first can run off down a very long (or infinite) path.

Perform depth first search but only to a pre-specified depth limit D.

-THE ROOT is at DEPTH 0, ROOT is a path of length 1.

-No node representing a path of length more than D+1 is placed on OPEN.

-We “truncate” the search by looking only at paths of length D+1 or less.

-Now infinite length paths are not a problem.

-But will only find a solution if a solution of DEPTH $\leq$ D exists.

Iterative-Deepening Search:
-Solves the problems of depth-first and breadth-first by extending depth limited search.
-Starting at depth limit L = 0, we iteratively increase the depth limit, performing a depth limited search for each depth limit.
-Stop if a solution is found, or if the depth limited search failed without cutting off any nodes because of the depth limit.
- If no nodes were cut off, the search examined all paths in the state space and found no solution then no solution exists.
Completeness:
-Yes, if a minimal depth solution of depth d exists.
Time Complexity: O(b^d) Most nodes lie on bottom layer.
-BFS can explore more states than IDS!
-In fact, IDS can be more efficient than breadth first search: nodes at limit are not expanded. BFS must expand all nodes until it expands a goal node. So the bottom layer it will add many nodes to OPEN before finding the goal node.
Space Complexity: O(bd)
-Will find shortest length solution which is optimal if costs are uniform.
- If costs are not uniform, we can use a “cost” bound instead.
- Only expand paths of cost less than the cost bound.
- Keep track of the minimum cost unexpanded path in each depth first iteration, increase the cost bound to this on the next iteration.
-This can be more expensive. Need as many iterations of the search as there are distinct path costs.
Path checking:
-Ensure that the state c is not equal to the state reached by any ancestor of c along this path.
-Paths are checked in isolation!
Cycle Checking:
-Keep track of all states previously expanded during the search.
-When we expand nk to obtain child c, ensure that c is not equal to **any previously** expanded state.
-This is called cycle checking, or multiple path checking.
-Higher space complexity (equal to the space complexity of breadth-first search.

Heuristic Search(informed search):

In uninformed search, we don't try to evaluate which of the nodes on OPEN are most promising. We never “look-ahead” to the goal.

-The idea is to develop a domain specific heuristic function h(n).

-h(n) guesses the cost of getting to the goal from node n (i.e., from the terminal state of the path represented by n).

heuristics are domain specific.

-If h(n1) < h(n2) this means that we guess that it is cheaper to get to the goal from n1 than from n2.

Greedy best-first search:

-We use h(n) to rank the nodes on OPEN

-Always expand node with lowest h-value.

-We are greedily trying to achieve a low-cost solution.

-However, this method ignores the cost of getting to n, so it can be lead astray exploring nodes that cost a lot but seem to be close to the goal.

A* search:

-Take into account the cost of getting to the node as well as our estimate of the cost of getting to the goal from the node.

Define an evaluation function f(n)

$f(n) = g(n) + h(n)$

-g(n) is the cost of the path represented by node n

-h(n) is the heuristic estimate of the cost of achieving the goal from n.

-Always expand the node with lowest f-value on OPEN.

-The f-value, f(n) is an estimate of the cost of getting to the goal via the node (path) n.

UNTIL PAGE 90!