In the name of God, the Compassionate, the Merciful

ARTIFICIAL INTELLIGENCE

Chapter 2: Intelligent Agents

- An **agent** is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuator.
- A **rational agent** chooses whichever action maximizes the expected value of the performance measure given the percept sequence to date.
- An agent is **autonomous** if its behaviour is determined by its own experience (ability to learn and adapt).
- To design a rational agent, we must specify the task environment: **PEAS** (Performance measure, Environment, Actuators, Sensors)

Environment types:

1. Fully observable (vs. partially observable)
2. Deterministic (vs. stochastic)
 - If the environment is deterministic except for the actions of other agents, then the environment is strategic.
3. Episodic (vs. sequential)
4. Static (vs. dynamic)
 - the environment is semi-dynamic if the environment itself does not change with the passage of time but the agents performance score does
5. Discrete (vs. continuous)
6. Single agent (vs. multiagent)

Agent types:

- **Look Up Table** (Dr.)
- Benefits: Easy to implement
- Drawbacks: Huge table, Take a long time to build the table, No autonomy, Even with learning, need a long time to learn the table entries
- **Simple reflex agents:** based on current percept ignoring percept history
- Selection based on condition-action rules.
- Advantage: Simplicity, requires only limited resources.
- Drawback: It only works if the environment is fully observable
- **Reflex agents with state (Model-based):** Agent uses model of the world around it to keep track of the parts of the worlds it can not always see
- **Goal-based agents**: Knowing about the current state of the environment is not always enough to decide what to do
- Goal information can ease the action selection process.
- Goal-based selection can be straightforward or can involve planning
- **utility-based agents**: Utility-related considerations can ease the selection of optimal action sequences.
- Utility function: ✓Maps state (sequence of states) into a real number ✓Resolves contradictions through trade-offs ✓Resolves uncertainty through measure for likelihood of success
- **Learning Agents**
  All these can be turned into learning agents

Chapter 3: Search

A **strategy** is defined by picking the order of node expansion

- Strategies are evaluated along the following dimensions:
- **Completeness:** Does it always find a solution if one exists?
- **Time complexity:** Number of nodes generated/expanded
- **Space complexity:** Maximum number of nodes in memory
- **Optimality:** Does it always find a least-cost solution?

Time and space complexity are measured in terms of (Dr.)

- **b:** Maximum branching factor of the search tree
- **d:** Depth of the least-cost solution
- **m:** Maximum depth of the state space (may be ∞)

Search Types:

- **Uninformed:** The agent has no information about the underlying problem other than its definition. e.g. Breadth-first, Uniform-cost, Depth-first, Depth-limited, Iterative deepening
- **Informed:** The agent have some idea of where to look for solutions

Tree search algorithms: offline, simulated exploration of state space by generating successors of already-explored states

- Breadth-first search(BFS):Expand shallowest unexpanded node.
- Uniform-cost search(UCS): Expand least-cost unexpanded node.
- Depth-first search(DFS): Expand deepest unexpanded node
- Depth-limited search(DLS): depth-first search with depth limit l
- Iterative deepening search(IDS):

Criterion:	Complete?	Time	Space	Optimal?
BFS	Yes[1]	$O(b^{d+1})$	$O(b^{d+1})$	Yes[2]
UCS	Yes[3]	$O(b^{\lceil \frac{c^*}{\epsilon} \rceil})$	$O(b^{\lceil \frac{c^*}{\epsilon} \rceil})$	Yes[4]
DFS	No[5]	$O(b^m)$	$O(bm)$	No
DLS	No	$O(b^l)$	$O(bl)$	No
IDS	Yes	$O(b^d)$	$O(bd)$	Yes

1. if b is finite
2. if cost = 1 per step
3. if step cost $\geq \epsilon$
4. nodes expanded in increasing order of $g(n)$
5. fails in infinite-depth spaces

Chapter 4: Informed search

Best-first search: Expand most desirable unexpanded node
Special cases: greedy search, A* search

Greedy best-first search: expands the node that appears to be closest to goal

A^* **Search:** avoid expanding paths that are already expensive
Evaluation function $f(n) = g(n) + h(n)$

- $g(n)$ = cost so far to reach n
- $h(n)$ = heuristic, estimated cost to goal from n
- $f(n)$ = estimated total cost of path through n to goal
- A^* search uses an **admissible** heuristic i.e., $h(n) \leq h^*(n)$ where $h^*(n)$ is the **true cost** from n.

Criterion:	Complete?	Time	Space	Optimal?
Greedy	No[1]	$O(b^m)$	$O(b^m)$	No
A^*	Yes	exp	all nodes	Yes

1. can get stuck in loops.

Local search algorithms

- When path doesn't matter (e.g., optimization problems).
- keep a single "current" state, try to improve it.
- **Two key advantages:**
  1. they use very little memory usually a constant amount.
  2. they can often find reasonable solutions in large or infinite (continuous) state spaces

Hill-climbing search problem: 1- Local maximum 2- Plateau (resulting in a random walk) 3- Ridges (slopes very gently toward a peak)

SOLUTION: Random restart hill-climbing

Simulated Annealing:

- Escape local maxima by allowing some "bad" moves but gradually decrease their **size** and **frequency**.
- Probability of a move decreases with the amount ΔE by which the evaluation is worsened.
- high T allows more worse moves, T close to zero results in few or no bad moves.
- One can prove: If T decreases slowly enough, then simulated annealing search will find a global optimum with probability approaching 1.
- Widely used in VLSI layout, airline scheduling, etc.

Local beam search:

- Keep track of k states rather than just one.
- Start with k randomly generated states.
- At each iteration, all the successors of all k states are generated.
- If any one is a goal state, stop; else select the k best successors from the complete list and repeat.

Genetic algorithms

- A successor state is generated by combining two parent states.
- Start with k randomly generated states (**population**).
- A state is represented as a string over a finite alphabet (often a string of 0s and 1s).
- Evaluation function (**fitness function**): Higher values for better states.
- Produce the next generation of states by **selection**, **crossover**, and **mutation**.

Chapter 5: CSP

Constraint satisfaction problems (CSPs)
- Backtracking search
- Local search

Example Problems:
- Map Coloring
- 8-queens
- Course/room scheduling
- Cryptarithmetic
- Line Labeling

General-purpose CSP algorithms use the graph structure

Standard search formulation (incremental)

- **Initial state**: the empty assignment
- **Successor function**: assign a value to an unassigned variable that does not conflict with current assignment
- **Goal test**: the current assignment is complete.

Backtracking search for CSPs

Backtracking search

- Depth-first search for CSPs with single-variable assignments is called backtracking search.
- Backtracking search is the basic uninformed algorithm for CSPs.

How to Improve

- Which variable should be assigned next?
- In what order should its values be tried?
- Can we detect inevitable failure early?

✓ **Minimum remaining values (MRV)**: choose the variable with the fewest legal values.
✓ **Most Constraining Variable (MCS)**: choose the variable with the most constraints on remaining variables
✓ **Least Constraining Value (LCS)**: To choose a value which puts minimum constraint on other variables.

Forward checking prevents assignments that guarantee later failure

Constraint propagation (e.g., **arc consistency**) does additional work to constrain values and detect inconsistencies.

Local search for CSPs

???

Chapter 6: Adversarial Search (Games)

"Game" in AI:

- A multi-agent, non-cooperative environment
- Zero Sum Result
- Turn Taking
- Deterministic
- Two Player

Games vs. search problems

- "Unpredictable" opponent → specifying a move for every possible opponent reply.
- Time limits → unlikely to find goal, must approximate

Minimax

- Perfect play for deterministic games
- Idea: choose move to position with highest **minimax value** = best achievable payoff against best play

Criterion:	Complete?	Time	Space	Optimal?
Minimax	Yes[1]	O(b^m)	O(bm)	Yes[2]

1. if tree is finite
2. against an optimal opponent

$\alpha - \beta$ pruning

- α: minimal score that player MAX is guaranteed to attain.
- β: maximum score that player MAX can hope to obtain.
- It is important to note that we now have a Max-Value Function and a Min-Value Function

Algorithm

1. Search down the tree to the given depth.
2. Once reaching the bottom, calculate the evaluation for this node.(i.e. it's utility).
3. Backtrack, propagating values and paths (according to what shown in next slides).
4. When the search is complete, the Alpha value at the top node gives the minimum score that the player is guaranteed to attain if using the path stored at the top node.

Properties of $\alpha - \beta$ pruning

- Pruning does not affect final result
- With "perfect ordering," time complexity = $O(b^{m/2})$

Chapter 7: Logical Agents

Logical agents apply inference to a knowledge base to derive new information and make decisions

Knowledge base: set of sentences in a formal language

Logic in general

- **Logics** are formal languages for representing information such that conclusions can be drawn.
- **Syntax** defines the sentences in the language.
- **Semantics** define the "meaning" of sentences (i.e., define truth of a sentence in a world).

Inference:

- **Entailment** means that one thing follows from another: KB $\models \alpha$.
- Entailment is a relationship between sentences (i.e., syntax) that is based on semantics.
- KB $\vdash_i \alpha$: i derives α from KB.
- An argument is **sound** if and only if 1-The argument is valid. 2-All of its premises are true.
- **Completeness** ???

Propositional logic is the simplest logic–illustrates basic ideas

P	Q	$P \vee Q$	$P \wedge Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
0	0	0	0	1	1
0	1	1	0	1	0
1	0	1	0	0	0
1	1	1	1	1	1

- $(P \Rightarrow Q) \equiv (\neg P \vee Q)$
- A sentence is **valid** if it is true in all models
- **Deduction Theorem:** KB $\models \alpha$ if and only if (KB $\Rightarrow \alpha$) is valid
- A sentence is **satisfiable** if it is true in some model.
- A sentence is **unsatisfiable** if it is true in no models.

Reasoning patterns (inference rules:)

- Modus Ponens $\frac{\alpha\Rightarrow\beta,\quad \alpha}{\beta}$
- And-Elimination, And-Introduction, Or-Introduction, Double Negation-Elimination
- Unit Resolution $\frac{\alpha\vee\beta,\quad \neg\beta}{\alpha}$
- Resolution $\frac{\alpha\vee\beta,\quad \neg\beta\vee\gamma}{\alpha\vee\gamma}$ or equivalently $\frac{\neg\alpha\Rightarrow\beta,\quad \beta\Rightarrow\gamma}{\neg\alpha\Rightarrow\gamma}$

Normal forms

- Conjunctive Normal Form (CNF):
$$(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$$
- Disjunctive Normal Form (DNF):
$$(A \wedge B) \vee (A \wedge \neg C) \vee (A \wedge B \wedge \neg C)$$
- Horn form: conjunction of Horn clauses (a clause (a disjunction of literals) with at most one positive literal).
$$(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$$

Proof methods

- Application of inference rules:
  - Legitimate (sound) generation of new sentences from old
  - a sequence of inference rule applications
  - Typically require transformation of sentences into a normal form
- Model checking:
  - truth table enumeration ($O(2^n)$ for n symbols;).
  - improved backtracking
  - heuristic search in model space

Resolution

- Resolution inference rule ???????????
- Resolution is sound and complete for propositional logic

Modus Ponens (for Horn Form): complete for Horn KBs

- Can be used with forward chaining or backward chaining.
- These algorithms are very natural and run in linear time

Forward vs. backward chaining:

- FC is data-driven, automatic, unconscious processing
- BC is goal-driven, appropriate for problem-solving
- Complexity of BC can be much less than linear in size of KB

Efficient propositional inference

Complete backtracking search algorithms

- DPLL algorithm (Davis, Putnam, Logemann, Loveland)
- Incomplete local search algorithms (e.g., WalkSAT algorithm)

Propositional logic has very limited expressive power

Chapter 8: First-Order Logic

First-order logic

- Whereas propositional logic assumes the world contains facts, first-order logic (like natural language) assumes the world contains:
  - **Objects:** people, houses, numbers, ...
  - **Relations:** red, round, prime, brother of, ...
  - **Functions:** father of, best friend, plus, ...

Chapter 9: Inference in first-order logic

References:

[1] Artificial Intelligence A Modern Approach, by Stuart Russell and Peter Norving

Made by ma.mehralian using LaTeX