1 Rational Agents

Agent: Perceives via sensors, acts via actuators
Rational: Maximizes expected performance given percepts and knowledge ($\neq$ omniscient)
Most challenging env: Partially observable, nondeterministic, strategic, dynamic, continuous, multi-agent

2 Search Fundamentals

Problem: State space, initial state, actions, goal test, path costs
Solution: Action sequence from initial $\rightarrow$ goal state

2.1 Search Criteria

Complete: Finds solution if exists
Optimal: Lowest path cost
Complexity: b (branching), d (goal depth), m (max depth)

2.2 Uninformed Search

BFS: layer by layer
UCS: Priority queue by path cost to node n: $g(n)$
DFS: deepest node first
IDS: DFS with increasing depth limits
Bidirectional: Forward+backward search

3 CSPs

Variables: $\{x_1, \ldots, x_n\}$, **Domains:** $\{dom_1, \ldots, dom_n\}$, **Constraints:** Allowable value combinations
Solution: Complete value assignment

3.1 Backtracking Heuristics

Most-Constrained Var: Fewest remaining values, red. b
Most-Constraining Var: Most constraints on unassigned, tie-breaker for (1)
Least-Constraining Value: Prefer value that rules out fewest choices for neighbors

3.2 Inference

Forward Checking: Delete inconsistent neighbor values, backtrack if any variables domain becomes empty
Arc Consistency (AC-3): $\forall x \in X, \exists y \in Y$ satisfying constraint, $O(d^3n^2)$, NP-hard

3.3 Exploiting Structure

Disconn. components of contraint graph can be solved independantly
Tree CSPs: $O(nd^2)$ with topological node order which enforces arc consistency (value assign. from root) (Almost-TCSPs: cutset conditioning (break loops in graph) and tree decomposition (connected subproblems organized as tree))

4 Informed Search

Heuristic $h(n)$ estimates cheapest cost to goal from node n
Consistent iff it is less than or equal to the actual cost of action a (overly optimistic) (implies admissibility)

4.1 Best-First Variants

Greedy: $f(n) = h(n)$, fast but incomplete/non-optimal
A*: $f(n) = g(n) + h(n)$ (UCS + Best-First)
- Complete/optimal if h admissible: $h(n) \leq h^*(n)$
- Graph-search optimal if h consistent, $h(s) - h(s') \leq c(a)$
- Exponential space complexity

IDA*: f-cost cutoff, memory efficient

4.2 Local Search (if goal path irrelevant)

Hill-climbing: Move to best neighbor state, issues: local maxima/plateaus/ridges
Simulated annealing: Accept bad moves with decreasing probability (temperature T)
Genetic algorithms: Population evolution: fitness func for individuals, selection, crossover, mutation

5 Games

Game: Initial state, operators (legal moves), terminal test, utility function (outcome of game)
Properties: States fully accessible, contingency problem, huge state space

5.1 Minimax

DFS game tree, MAX maximizes, MIN minimizes utility
Use evaluation function for non-terminals (prefer quiescent positions) to avoid horizon effect

5.2 Alpha-Beta Pruning

α: best MAX value, β: best MIN value so far
Prune when $\alpha \geq \beta$ (both MIN and MAX), Best case: $b \rightarrow \sqrt{b}$ (depends on move ordering)

5.3 Chance Games

Add chance nodes: value $= \sum P(\text{outcome}) \times \text{value(outcome)}$

6 Propositional Logic

Syntax: Literals, clauses (disjunctions), **Semantics:** Truth interpretations **Entailment:** KB $\models \alpha$ if α true in all KB models

6.1 CNF Conversion

1. Eliminate $\Rightarrow, \Leftrightarrow$: $\alpha \Rightarrow \beta$ becomes $(\neg\alpha \vee \beta)$
2. Move $\neg$ inward: $\neg(\alpha \wedge \beta)$ becomes $(\neg\alpha \vee \neg\beta)$
3. Distribute $\vee$ over $\wedge$: $(\alpha \wedge \beta) \vee \gamma$ becomes $(\alpha \vee \gamma) \wedge (\beta \vee \gamma)$

CNF: $\bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{m_i} l_{i,j} \right)$, **DNF:** $\bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{m_i} l_{i,j} \right)$

6.2 Resolution (derive formulae from KB)

Goal: Prove KB $\models \alpha$ by showing KB $\cup\{\neg\alpha\}$ unsatisfiable
Req: All sentences in CNF
Rule: From $C_1 \cup \{l\}, C_2 \cup \{\neg l\}$ derive resolvent $C_1 \cup C_2$
Empty clause $\square$ proves unsatisfiability

7 Boolean Satisfiability Problem (SAT)

Goal: Find satisfying assignment (model) for CNF formula or prove none exists.

7.1 DPLL

Given a set of clauses Δ over a set of variables Σ:
1. If $\Delta = \emptyset$ return "satisfiable"
2. If $\square \in \Delta$ return "unsatisfiable"
3. Unit propagation: assign unit clause literals, recurse
4. Split: try both values for unassigned variable, recurse

7.2 CDCL

Enhances DPLL with conflict analysis, clause learning and backjumping.

8 Action Planning

Goal: Find action sequence to achieve goals.

8.1 STRIPS

States: Sets of true propositions (closed world assumption)
Actions: Preconditions + effects (add/delete lists)
Planning task: $\langle S, O, I, G \rangle$ (states, operators, initial, goal)

8.2 PDDL

Standard planning language, extends STRIPS with typing, conditional effects, numerical resources

8.3 Algorithms

Progression: Forward search from initial state
Regression: Backward search from goal

9 Probability / Decisions

Prior: $P(A)$, **Posterior:** $P(A|B)$
Product rule: $P(A \wedge B) = P(A|B)P(B)$
Independence: $P(a|b) = P(a)$ iff $P(a \wedge b) = P(a)P(b)$
Bayes' Rule:

$$P(\text{cause}|\text{effect}) = \frac{P(\text{effect}|\text{cause})P(\text{cause})}{P(\text{effect})}$$

9.1 Bayesian Networks

Structure: Directed Acyclic Graph, nodes = variables, edges = dependencies
Joint distribution: $P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|\text{parents}(x_i))$
Inference: Generally NP-hard, $O(2^n)$. Linear for polytrees.
Conditional Independence: Node independent of non-descendants given parents

10 (Action) Decision Theory

Utility: $U(S)$ assigns values to states
Axioms: Orderability, Transitivity, Continuity
MEU: $EU(A|E) = \sum_i P(\text{Result}(A) = i|\text{Do}(A), E)U(i)$ (rational agent maximizes MEU)

10.1 MDPs

Components: States S, actions A, transitions $P(s'|s, a)$, rewards $R(s)$
Policy: $\pi(s)$ maps states $\rightarrow$ actions (goal: find $\pi*$)

10.2 Bellman Equation

$U(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s, a)U(s')$

$\pi(s) = \arg\max_a \sum_{s'} P(s'|s, a)U(s')$

10.3 Algorithms

Value iteration: $U'(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} P(s'|s, a)U(s')$
Policy iteration: Alternate evaluation + improvement

11 Machine Learning

Ockhams razor: Choose simplest hypothesis consistent with the data

11.1 Decision Trees

Algorithm: Greedy divide-and-conquer
1. Same class $\rightarrow$ leaf
2. Select best attribute (max info gain)
3. Recurse

- **Entropy:** $I(P(y_1), \ldots, P(y_n)) = \sum_{i=1}^{n} -P(y_i) \log_2 P(y_i)$
- **Evaluation:** Separate training/test sets

12 Deep Learning

Multiple layers learn feature hierarchies

12.1 MLPs

Structure: Input, hidden, output layers **Training:** Minimize loss with SGD + backpropagation

13 Quick Reference

13.1 Search Complexities

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^l)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(bl)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

b branching factor
d depth of solution
m maximum depth of the search tree
l depth limit
C^* cost of the optimal solution
ϵ minimal cost of an action

Superscripts:
[a] b is finite
[b] if step costs not less than ϵ
[c] if step costs are all identical
[d] if both directions use breadth-first search

13.2 Probability Rules

$P(A \cup B) = P(A) + P(B) - P(A \cap B)$
$P(A|B) = \frac{P(A \cap B)}{P(B)}$ if $P(B) > 0$

13.3 Logic Equivalences

$\neg(A \wedge B) \equiv (\neg A \vee \neg B)$ (De Morgan)
$\neg(A \vee B) \equiv (\neg A \wedge \neg B)$ (De Morgan)
$(A \Rightarrow B) \equiv (\neg A \vee B)$
$(A \Leftrightarrow B) \equiv ((A \Rightarrow B) \wedge (B \Rightarrow A))$

13.4 AC-3 Step-by-Step

For each arc (X_i, X_j): Remove values from $dom(X_i)$ that have no supporting value in $dom(X_j)$. If domain becomes empty, return inconsistent.

13.5 CNF Conversion Examples

$\phi \Leftrightarrow \psi \equiv (\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi) \equiv (\neg\phi \vee \psi) \wedge (\neg\psi \vee \phi)$

13.6 Resolution Steps

To prove $KB \models \alpha$ show that $K \cup \{\neg\alpha\} \models \bot$:
1. Add $\neg\alpha$ to KB
2. Convert all to CNF clauses
3. Apply resolution rule until $\square$ derived
4. If $\square$ found $\Rightarrow$ KB $\models \alpha$

13.7 MDP Value Update

$U^{t+1}(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s, a)U^t(s')$

13.8 MDP Policy Improvement

1. Policy evaluation, given policy π_t calc. utilities $U_t = U^{\pi_t}$
2. Policy improvement

$$\pi_{t+1}(s) = \text{argmax}_a \sum_{s'} P(s'|s, a)U_t(s')$$

13.9 Probability Calculations

$P(A \cup B) = P(A) + P(B) - P(A \cap B)$
$P(A|B) = \frac{P(A \cap B)}{P(B)}$
If mutually exclusive: $P(A \cap B) = 0$
If independent: $P(A|B) = P(A)$

13.10 Information Gain Calculation

$Gain(A) = Entropy(S) - \sum_v \frac{|S_v|}{|S|} Entropy(S_v)$
where S_v = subset with attribute $A = v$
Entropy: $H(S) = -\sum_i p_i \log_2 p_i$

13.11 Alpha-Beta Pruning Rules

Initialize root with $\alpha = -\inf$, $\beta = \inf$
Prune if $\beta \leq \alpha$
Update: $\alpha = \max(\alpha, value)$ at MAX
Update: $\beta = \min(\beta, value)$ at MIN

13.12 CSP Arc Consistency

Arc $X \rightarrow Y$ consistent if: $\forall x \in D_X, \exists y \in D_Y$ such that (x, y) satisfies constraint

13.13 Minimax with Chance

Expected value at chance nodes: $V = \sum P(outcome_i) \times V(child_i)$

13.14 A* Completeness

Finite state space, non-negative edge costs

13.15 Misc

If two events A and B are unconditionally independent, they may not be conditionally independent given another event C.

If two DTs are equally accurate, the one with fewer nodes is preferrable since it is less prone to overfitting.

ML: Manual feature engineering, simpler models, perform well on structured data and simpler problems, requires less computational power

DL: Automatically learns features from data, can handle large complex datasets by learning hierarchical features and using deep architectures, requires a lot more computational power