Problem Environments Properties (not related to agents):
Fully/Partially observable   Deterministic/Stochastic   Episodic/Sequential
Discrete/Continuous   Single/Multi agent   Known/Unknown Static/Dynamic

• Uninformed Search

State (initial): s_i/s_0 Goal test: $\text{isGoal} : s_i \to \{0,1\}$ Action: $\text{actions} : s_i \to A$
Action cost: $\text{cost} : (s_i, a_j, s_i') \to v$ Transition model: $\text{T} : (s_i, a_j) \to s_i'$
Performance Criteria: Time / Space / Completeness / Optimality

```
frontier = {initial state} # data structure
while frontier:
  current = frontier.pop()
  if isGoal(current): return True
  for a in actions(current): frontier.push(T(current, a))
return False
```

b: branch factor d: depth for sol m: max depth ℓ: depth limit e: $1 + \lfloor C^*/\epsilon \rfloor$

Algo	Time	Space	Complete	Optimal
BFS(Q)	b^d	b^d	✅ if b finite && (d finite || has sol)	❎ unless unif cost
UCS(PQ)	b^e	b^e	✅	✅
DFS(S)	b^m	bm	✅ if finite	❎
DLS(S)	b^ℓ	$b\ell$	❎	❎
IDS(S)	b^d/b^m no sol	bd/bm no sol	✅ (BFS)	❎ (BFS)

• Informed Search

Heuristic Function:
Admissibility: $\forall n, h(n) \leq h^*(n),\ h(G) = 0$, consistency $\to$ admissibility
Consistency: monotonically increasing, $\forall n, h(n) \leq \text{cost}(n, a, n') + h(n')$
Dominance: $\forall n, h_1(n) \geq h_2(n)$

Greedy Best-First Search: $f(n) = h(n)$, not optimal, time and space $O(b^m)$

Tree search not complete can stuck in loop, graph search complete if finite

A* Search: $f(n) = g(n) + h(n)$

Tree search optimal if $h(n)$ admissible, graph search optimal if $h(n)$ consistent

• Local Search

$O(b)$ space, applicable to large/infinite state space, but incomplete

1. Hill-Climbing Search

```
current = initial state
while True:
  neighbour = highest-valued successor of current
  if value(neighbour) <= value(current): return current
  current = neighbour
```

`value`: $f(n) = -h(n)$, can use $f(n) = h(n)$ by replacing `<=` with `>=`

Issues: can get stuck at local maxima / ridges / plateaus / shoulders

Variants:

Sideways move: `value(nb) < value(curr)` in hope that plateau is a shoulder

Stochastic: Choose uphill moves at random

First-choice: Generate successors randomly until better value

Random restart: Keep attempting from random initial states until goal is found

Computation: $\text{steps for success} + \frac{1-p}{p} \times \text{steps for failure}$

2. Local Beam Search Performs better than k random restarts in parallel

Variants: Stochastic beam search

• Constraint Satisfaction Problems

Formulation:

$\mathcal{X}$: variables $\mathcal{D}$: domains, $D_i = \{v_1, \ldots, v_k\}$ $\mathcal{C}$: constraints, $C_j = \langle \text{scope}, \text{rel} \rangle$

Initial state: all variables unassigned Goal state: complete and consistent

Complete: all variables assigned Consistent: no constraint violated

Constraint Graph: Vertex: variable Link: global constraint Edge: relation

Search Tree Size:

Assume D same for all domains, order of assignment not important

Branching factor at depth ℓ : $(|\mathcal{X}| - \ell) \cdot |D|$ states

#leaf: $nm \times (n-1)m \times \cdots \times m = n!m^n$, where $n = |\mathcal{X}|$ and $m = |D|$

AC-3:

Art-consistency: X_i AC w X_j iff $\forall x \in D_i \, \exists y \in D_j$ s.t. (X_i, X_j) satisfied
Maintains queue of arcs, pops arbitrary (X_i, X_j) to check arc consistency
If D_i unchanged, move onto the next arc
If D_i revised, enqueue all (X_k, X_i) where X_k is a neighbour of X_i
$|\mathcal{X}| = n \to O(n^2)$ arcs, $|\mathcal{D}| = d \to (X_i, X_j)$ enqueued d times as $|D_j| = d$
Checking consistency: $O(d^2)$ Time: $O(n^2 d^3)$

Backtracking Search:

I. select-unassigned-variable: variable-order heuristics
  a. <u>most-remaining-values</u> / most-constrained-variable / fail-first
    Pick X with smallest $|D|$
  b. <u>degree</u>: Tie-breaker for MRV, pick X that restricts most # unassigned X

II. order-domain-values: value-order heuristics
  a. <u>least-constraining-value</u> / fail-last
    Pick value that rules out fewest choices for neighbouring X

III. inference: determine if it leads to terminal state
  a. forward checking: terminate when any X has $|D| = 0$, no early detection
  b. maintaining arc consistency
    After each assignment only enqueue unassigned neighbours

IV. backtrack: restore previous state

• Adversarial Search

$\text{toMove} : s \to p$: next player to move $\text{isTerminal} : s \to \{0, 1\}$: terminal test
$\text{actions} : s_i \to A$: legal moves $\text{utility} : (s, p) \to v$: for player p at terminal s
$\textbf{MiniMax}(s)$: DFS, complete if game tree finite, $O(b^m)$ time, $O(bm)$ space

$$\begin{cases} \text{utility}(s, \text{toMove}(s)) & \text{if isTerminal}(s) \\ \max_{a \in \text{actions}(s)} \text{MiniMax}(\text{result}(s, a)) & \text{if toMove}(s) = \text{MAX} \\ \min_{a \in \text{actions}(s)} \text{MiniMax}(\text{result}(s, a)) & \text{if toMove}(s) = \text{MIN} \end{cases}$$

***α-β* Pruning**: additional layer for MiniMax
  Maintain α of MAX initially $-\infty$, β of MIN initially $+\infty$, prune if $\alpha \geq \beta$
  Perfect ordering: $O(b^{\frac{m}{2}})$ Random ordering: $O(b^{\frac{3}{4}m})$

$\textbf{Heuristic-MiniMax}(s)$: cut off search early and apply eval function
$\text{isCutoff} : s \to \{0, 1\}$: true for terminal states $\text{eval} : (s, p) \to v$: eval function

$$\begin{cases} \text{eval}(s, \text{toMove}(s)) & \text{if isCutoff}(s, d) \\ \max_{a \in \text{actions}(s)} \text{H-MiniMax}(\text{result}(s, a), d + 1) & \text{if toMove}(s) = \text{MAX} \\ \min_{a \in \text{actions}(s)} \text{H-MiniMax}(\text{result}(s, a), d + 1) & \text{if toMove}(s) = \text{MIN} \end{cases}$$

• Logical Agents

KB-based agents tell KB percept, ask KB for action, tell KB chosen action

Entailment: $\alpha \models \beta \Leftrightarrow M(\alpha) \subseteq M(\beta)$

Inference Algorithm: $\mathrm{KB} \vdash_{\mathcal{A}} \alpha$: α derived from KB by algorithm $\mathcal{A}$

I. Soundness: $\mathrm{KB} \vdash_{\mathcal{A}} \alpha \Rightarrow \mathrm{KB} \models \alpha$, $\mathcal{A}$ will not infer nonsense

II. Completeness: $\mathrm{KB} \models \alpha \Rightarrow \mathrm{KB} \vdash_{\mathcal{A}} \alpha$, $\mathcal{A}$ infers any sentence KB entails

Truth Table Enumeration: n variables, 2^n assignments

Enumerate all models, check if $M(\mathrm{KB}) \subseteq M(\alpha)$, DFS, $O(2^n)$ time, $O(n)$ space

Sound since follow entailment definition, complete as finite models to check

CNF: $\Leftrightarrow \rightarrow\ (\Rightarrow) \wedge (\Rightarrow)$ | $\Rightarrow \rightarrow\ \neg\vee$ | move $\neg$ inwards | $\vee(\wedge) \rightarrow (\vee) \wedge (\vee)$

Validity: true for all assignments **Satisfiability**: true for some assignments

Inference Rules: Modus Ponens, $\alpha \wedge \beta \Rightarrow \alpha$, 1231, resolution

Resolution Algorithm: show $\mathrm{KB} \wedge \neg\alpha$ is unsatisfiable to prove $\mathrm{KB} \models \alpha$

I. Convert $\mathrm{KB} \wedge \neg\alpha$ into CNF

II. Resolve clauses until empty clause ($\mathrm{KB} \models \alpha$) or no more resolution ($\mathrm{KB} \not\models \alpha$)

Sound as steps use sound inference rules, complete resolution closure

• Uncertainty

W/O cond indep: $2^n - 1$ entries W cond indep: $n + 1$ entries

Bayes' Rule: $P(\mathrm{Cause}|E_1, \ldots, E_k) = \alpha \cdot \prod_i P(E_i|\mathrm{Cause}) \cdot P(\mathrm{Cause})$

Bayesian Network:

I. Nodes represent random variables

II. DAG, $X \rightarrow Y \Rightarrow X$ directly influences Y

III. $P(X|\mathrm{Parents}(X))$ for each node in conditional probability table

Relationships:

Indep events: $P(A \wedge B \wedge C) = P(A) \cdot P(B) \cdot P(C)$

Indep causes: $P(A \wedge B \wedge C) = P(C|A, B) \cdot P(A) \cdot P(B)$

Cond indep effects: $P(A \wedge B \wedge C) = P(C|A) \cdot P(B|A) \cdot P(A)$

Casual chain: $P(A \wedge B \wedge C) = P(C|B) \cdot P(B|A) \cdot P(A)$

Bayes Net Construction Algorithm:

I. Choose an ordering $\{X_1, \ldots, X_n\}$

II. For $i = 1, \ldots, n$:

a. Select minimal parent set s.t. $P(X_i|\mathrm{Parents}(X_i)) = P(X_i|X_1, \ldots, X_i - 1)$

b. Link every parent to X_i

c. Write down CPT for $P(X_i|\mathrm{Parents}(X_i))$

Network constructed is acyclic, contains no redundant probability values