IS-ZC444: ARTIFICIAL INTELLIGENCE

Lecture-03: Intelligent Agents (Contd..), Problem Solving by Search

Dr. Kamlesh Tiwari
Assistant Professor
Department of Computer Science and Information Systems,
BITS Pilani, Pilani, Jhunjhunu-333031, Rajasthan, INDIA

Sept 05, 2020 FLIPPED (WILP @ BITS-Pilani Jul-Nov 2020)

Intelligent Agents

In pursuit of computers doing things which at the moment, people do better, AI attempts to build intelligent entities called **Agents** [1]

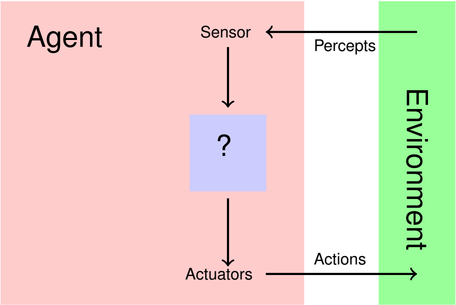
Agent perceives the environment through **sensors** and act upon the environment through **actuators**

- Our approach is to build **rational agent**
- How well agent can behave depends on the nature of environment. Some environments are more difficult.

Agents choice of action can depend on percept sequence.

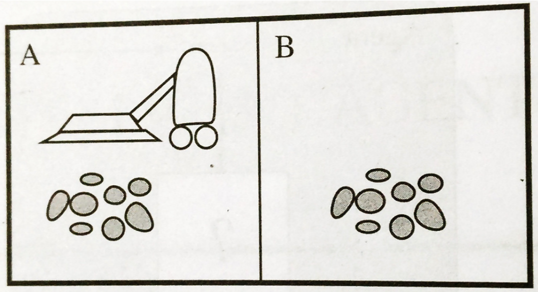
[1] Consider human, robot or software agent

Intelligent Agents



- Agent may use entire percept sequence to choose action
- Mathematically, **agent function** maps the percept sequence to an action. Tabulation of the function is hard due to number of states
- **Agent program** is the logic implemented in physical system

Example: Toy Vacuum Cleaner



Two cells, dirt/or-not. Can sense dirt and move

- If current sequence is dirty, then suck; otherwise move to other square.
- What makes this agent intelligent?

Tabulation of Vacuum Cleaner World

Percept	**Action**
[A,Clean]	Right
[A,Dirty]	Suck
[B,Clean]	Left
[B,Dirty]	Suck
⋮	
[A,Clean], [A,Clean]	Right
[A,Clean], [A,Dirty]	Suck
⋮	

- Different agents could be defined by filling in the right-hand side column of the table in various way. ($actions^{historysize \times perceptStates}$)
- Question remains which agent is better

Rational Agent

Rational Agent is one that does the right things

- Sequence of actions of agent leads to sequence of *states of the environment*
- A **performance measure** could be used to evaluate how the sequence of state of environment is desirable (NOT of agent)
- Design performance measure according to what one actually wants in the environment, rather then how agent should behave
- What is desired is not easy to define (simple life or ups and down) (every one in moderate poverty or some rich some more poor)

Rationality depends on

1. Performance measure that defines the criteria of success
2. Agents prior knowledge of the environment
3. The action that the agent can perform
4. Agents percept sequence till date

Rational Agent

For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

Two performance measure for our Vacuum Cleaner agent

- Rational if: awards one point for clean square at each time stamp
- Not rational if: awards one point for clean square at each time stamp and puts penalty for movement to left or right

Rationality is not **perfection**. Rationality maximizes expected performance whereas perfection maximizes actual performance.

Omniscience, Learning and Autonomy

- Omniscient agent knows actual outcome of its actions (this is impossible)
- Information gathering is an important part of rationality (agent should get appropriate percept before taking action)
- Agent initial configuration could reflect some prior *knowledge*. The agent can modify it and augment
- In some cases knowledge about environment states could be there priory
- An **autonomous agent** learns to compensate for partial or incorrect prior knowledge

Task environment is the "problem" which the rational agent have to "solve"

PEAS: Task Environment

Performance, **E**nvironment, **A**ctuators, **S**ensors (PEAS) formally describes the task environment

Consider Autonomous Taxi

- **Performance Measure**: safe, fast, legal, confortable, maximize profit
- **Environment**: roads, traffic, pedestrians, customers
- **Actuators**: Steering, accelerator, brake, signal, horn, display
- **Sensors**: Camera, sonar, speedometer, GPS, odometer, engine sensors, keyboard

Software agents (softbots) exists in rich and unlimited domain

PEAS: Examples

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical Diagnostic System	Healthy patient, Reduced cost	Patient, Hospital, Staff	Display of questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's answer
Satellite image analysis system	Correct image categorization	Downlink from orbiting satellite	Display of scene categorization	Color pixel array
Part-picking robot	percentage of parts in correct bins	Conveyor belt with parts, bins	Joined arm and hands	Camera, joint angle sensors
Refinery controller	Purity, yield, safety	Refinery, operators	Values, pumps, heaters, displays	Temperature, pressure, chemical sensors
Interactive English tutor	Student's score on test	Set of students, testing agency	Display of exercises, suggestions, corrections	Keyboard entry

Task Environment Properties

- **Fully Observable** vs **Partially Observable**
- **Single agent** vs **Multi agent**: competition, cooperation, communication
- **Deterministic** vs **Stochastic**: next state is uncertain, non-deterministic
- **Episodic** vs **Sequential**: uncertain, non-deterministic
- **Static** vs **Dynamic**
- **Discrete** vs **Continuous**
- **Known** vs **Unknown**

Task Environment: Examples

Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Taxi driving	Partially	Multi	Stochastic	Sequential	Dynamic	Continuous
Medical diagnostic	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continuous
Part-picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continuous
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Interactive English Tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete
Backgammon	Fully	Multi	Stochastic	Sequential	Static	Discrete

Structure of Agent

Job of AI is to design an agent program that implements the agent function.

$$agent = archiecture + program$$

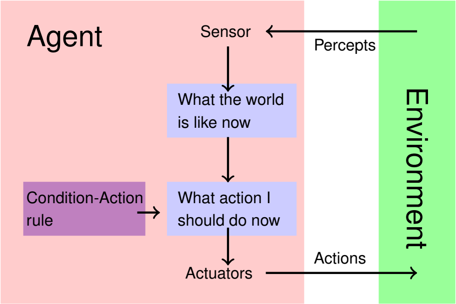
Table driven agent

Algorithm 1: Table-Driven-Agent (percept)

append *percept* to *percepts*
action = LOOKUP(*percepts*,table)
return action

- Because of combinatorics, size of the table is an issue. Also it would be not be a nice idea to have table.
- Four basic kind of agents are: Simple reflex, model based, goal based, and utility based

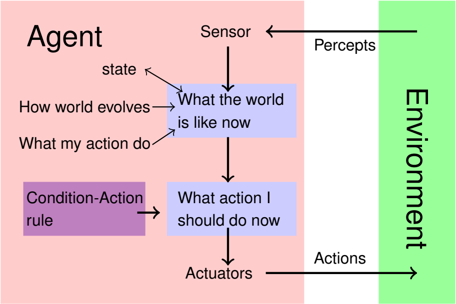
Simple Reflex Agent



Algorithm 2: Simple-Reflex-Agent (percept)

1. state = INTERPRET-INPUT(*percept*)
2. rule = RULE-MATCH(*state*,*rules*)
3. action = rule.ACTION
4. **return** action

Model Based Reflex Agent

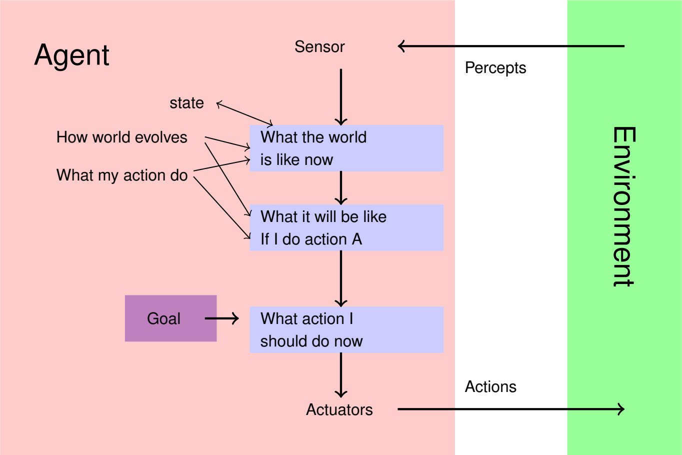


Algorithm 3: Model-Based-Reflex-Agent (percept)

---

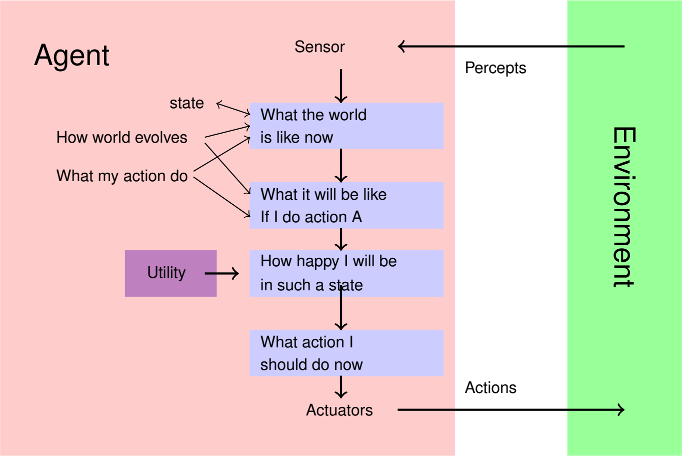
1 state $=$ UPDATE-STATE(*state*,*action*,*percept*,*model*)
2 rule $=$ RULE-MATCH(*state*,*rules*)
3 action $=$ rule.ACTION
4 **return** action

Goal Based Agent



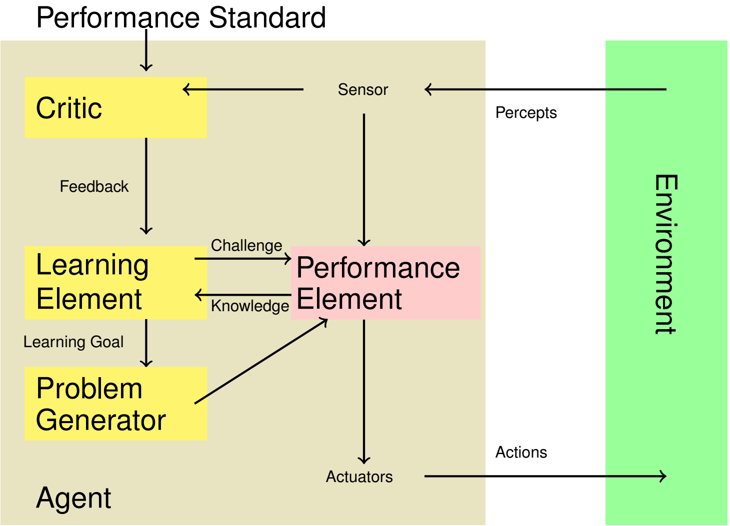
Search and **Planning** is needed. Consideration of future is important. Adaptive behavior change is possible.

Utility Based Agent



Utility evaluate how good it is. How cheap it is to reach the goal. Maximize **expected utility**. Trade-off between objectives could be managed. Can handle uncertainty.

A General Learning Agent



Critic finds goodness of agent. **Learning element** make rules to improve or adapt. **Problem Generator** suggest experiments to test

Problem-Solving Agent

Reflex agents cannot operate well if needs planning (or large table).

- A goal-based agent called **problem-solving agent** uses state of the world (cumulative)
- Uninformed search algorithms are not given any information about the problem other than its definition. They work, but may not efficiently (performance measure is always a concern for being intelligent).

Consider an agent enjoying holiday in Arad, Romania

What are performance measure? improve suntan, improve Romanian, sight seeing *etc*.

What if he has a flight from Bucharest next day?
Adopt the goal of getting Bucharest on time.

Problem-Solving Agent

- **Goal** is some world-state.
- Agent's task is to find out how to act[2] now and in future.
- In our example[3] let three roads lead out of Arad, one towards Sibu, one to Timisoara, and one to Zerind. None of these achieves the goal. (he needs some familiarity with the geography of Romania *i.e.* environment)
- A map can specify the environment.
- Here environment is
  observable (agent always knows his current state)
  discrete (agent have finitely many actions to take)
  known (agent knows which action takes to which state)
  deterministic (each action has only one outcome)

Here solution to a problem corresponds to a fixed sequence of actions.

[2] Actions could be abstract, like goto A to B (not move 5 step, rotate 10 degree ...)
[3] Agent want to go to Bucharest

Search

- The process of looking for a sequence of actions that reaches to goal is **search**
- How to look for, is an important question

Problem is defined using five components

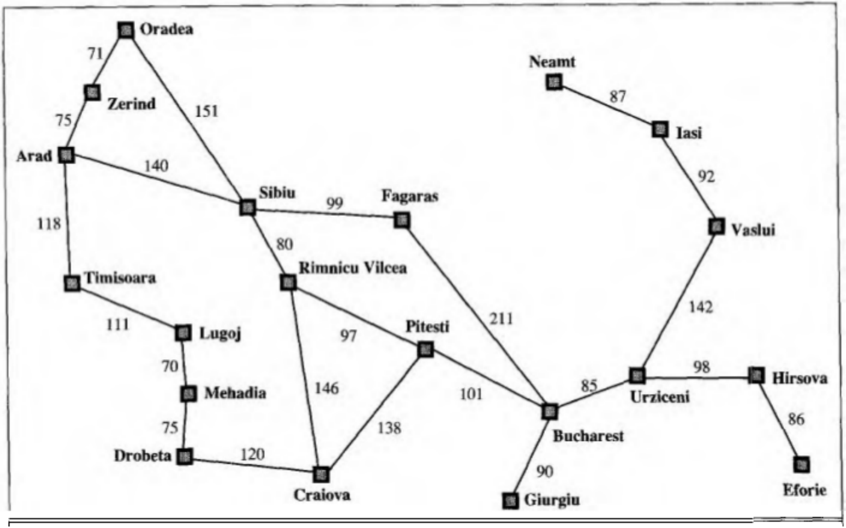
1. The **initial state**: $in(Arad)$
2. Set of **actions**: $\{go(Subiu), go(Timisoara), go(Zerind)\}$
3. **Transition model**: $Result(in(Arad), go(Zerind)) = in(Zerind)$
4. **Goal test**: $\{in(Bucharest)\}$
5. **Path cost**: used to determine efficiency

Problem-Solving Agent

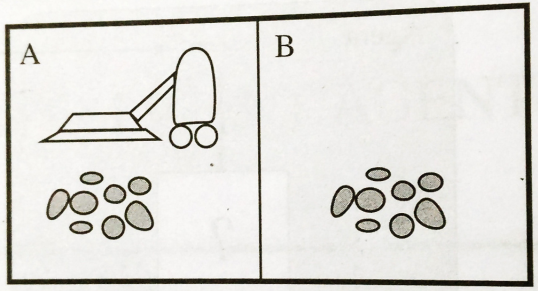
```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
   inputs: percept, a percept
   static: seq, an action sequence, initially empty
           state, some description of the current world state
           goal, a goal, initially null
           problem, a problem formulation

   state ← UPDATE-STATE(state, percept)
   if seq is empty then do
      goal ← FORMULATE-GOAL(state)
      problem ← FORMULATE-PROBLEM(state, goal)
      seq ← SEARCH(problem)
   action ← FIRST(seq)
   seq ← REST(seq)
   return action
```

Problem-Solving Agent



Problem-Formulation: Toy Vacuum Cleaner



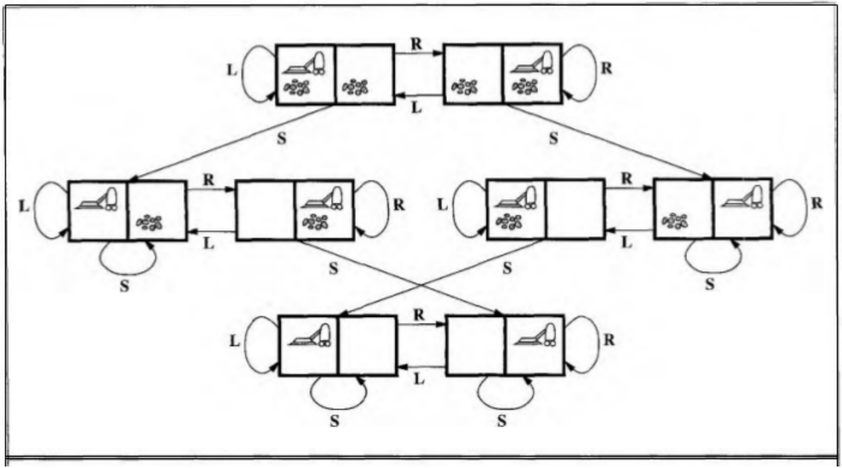
Two cells, dirt/or-not. Can sense dirt and move

- Move L and R, and suck S
- How many states? (room A/B, noDirt/oneRoomDirt/twoRoomDirt)
  $2 \times 2^2 = 8$

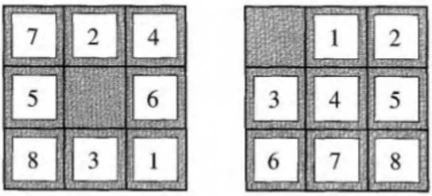
Determine the state space.

Problem-Formulation: Toy Vacuum Cleaner

The state space..



Problem-Formulation: 8-Puzzle

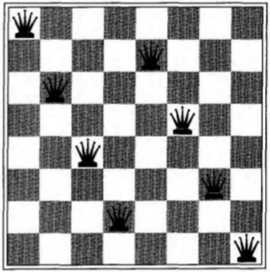


Start State Goal State

Determine

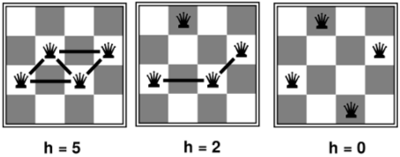
- States? is it 9!
- Initial State
- Actions
- Transition Model
- Goal Test
- Path cost

Problem-Formulation: 8-queens



Determine

- States
- Initial State
- Actions
- Transition Model
- Goal Test



Problem-Formulation: Knuth conjuncture

Using only a number 4, one can reach to any desired positive number, by applying a sequence of factorials, square root and floor operation

$$\left\lfloor \sqrt{\sqrt{\sqrt{\sqrt{\sqrt{(4!)!}}}}} \right\rfloor = 5$$

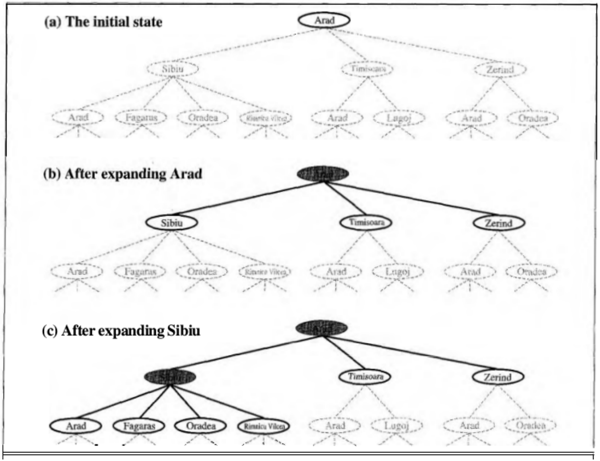
Determine

- States
- Initial State
- Actions
- Transition Model
- Goal Test

More Problems (real world)

- Route finding
- Touring problem: visit each city
- TSP: touring with single visit of cities
- VLSI layout
- Robot navigation
- Automatic assembly sequencing

Searching for Solution: search tree

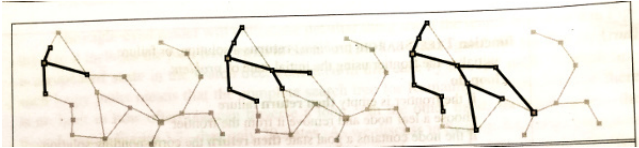


Searching for Solution: Algorithm

function TREE-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

---

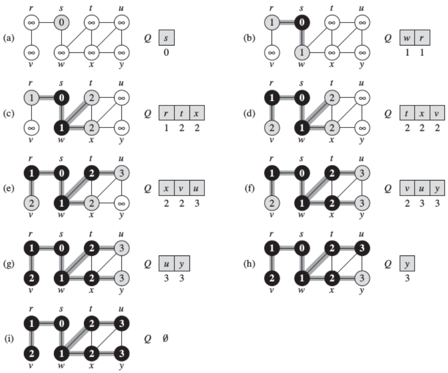
function GRAPH-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  initialize the explored set to be empty
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    add the node to the explored set
    expand the chosen node, adding the resulting nodes to the frontier
      only if not in the frontier or explored set



Uninformed Search Strategies (blind search): BFS

Breadth-first search root node is expanded first then all its successors.

```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

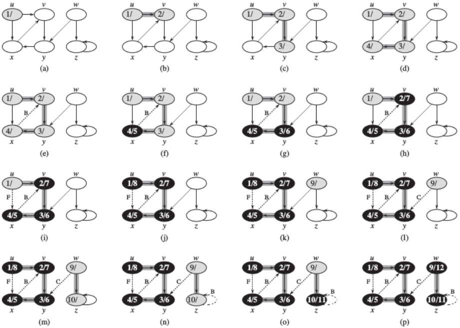


Uninformed Search Strategies (blind search): DFS

Depth-first search goes deep to branches first

```
DFS(G)
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT(G, u)

DFS-VISIT(G, u)
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-VISIT(G, v)
u.color = BLACK
time = time + 1
u.f = time
```



Thank You!

Thank you very much for your attention!

Queries ?

(Reference[4])

[4] Book - *AIMA*, ch-03, Russell and Norvig., and Book - Introduction to Algorithms by Cormen ch-22