Problem Solving

Introduction

A problem is a situation which is experienced by an agent and is solved by a sequence of actions that reduce the difference between the initial situation and the goal.

Components of a problem

- Initial state

  The state from which the agent starts to solve a problem
- Actions

  Description of possible actions in particular state
- Transition Model

  Description of each action with respect to state
- Goal Test

  Determination of the particular state is goal or not
- Path Cost

  Calculates the cost to each path

Defining problem as a state space search

A set of initial state, actions and the goal state for a given problem is known as state space for that problem. A state space represents a problem in terms of states and operators that changes the states. A state space essentially consists of a set of nodes representing each state of the problem, arcs between nodes representing the legal moves from one state to another, an initial state and a goal state

The major components of state space representation are:

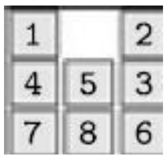
- Initial state
- Goal state, and
- Operator or legal moves.

Q. Explain problem as a state space with an example.

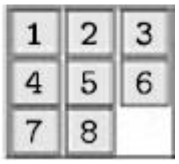
To explain problem as a state space representation in more detailed manner, let us consider a problem of 8puzzle game. The puzzle consists of an 8-square frames and an empty slot. The tiles are numbered from 1-8. It is possible to move the tiles in the square field by moving tile into the empty slot. The objective is to get square in a numeric order.

Here,

- **Initial State**



- **Operators**
  - UP: If the empty slot is not touching the up-frame, move it up.
  - DOWN: If the empty slot is not touching down-frame, move it down.
  - LEFT: If the empty slot is not touching left-frame, move it left.
  - RIGHT: If the empty slot is not touching right-frame, move it right.
- **Final State**



Problem Formulation and Problem types

The process of deciding the initial states to consider for the particular problems so that the operator perform the specified operation to reach the given a goal is called problem formulation. All the components of problems form a problem formulation and this formulation is abstract.

Toy problems

Toy problems illustrate various problem solving methods with exact description and can be used to compare performance.

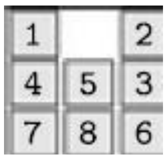
Examples: 8 – Puzzle game, 8 – Queens Problems, Tic - Tac Toe

Q. 8 Puzzle Game

The puzzle consists of an 8-square frames and an empty slot. The tiles are numbered from 1-8. It is possible to move the tiles in the square field by moving tile into the empty slot. Manage the tile square in a numeric order.

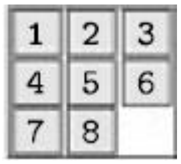
Here,

- **Initial State**

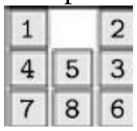


- **Operators**

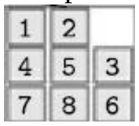
– UP: If the empty slot is not touching the up-frame, move it up.
– DOWN: If the empty slot is not touching down-frame, move it down.
– LEFT: If the empty slot is not touching left-frame, move it left.
– RIGHT: If the empty slot is not touching right-frame, move it right.

- **Final State** 

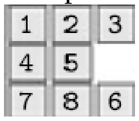
- **Solution**
  - Step 1



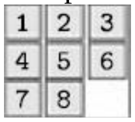
- Step 2



- Step 3



- Step 4

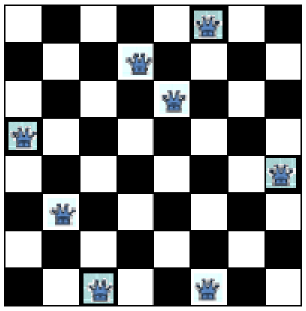


Q.8 queen's problem

The problem is to place 8 queens on a chessboard so that no two queens are in the same row, column or diagonal

Here,

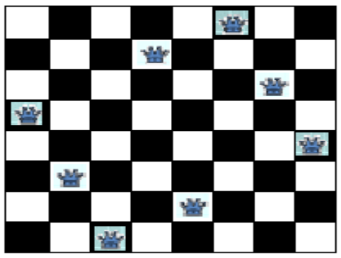
- **Initial State**



- **Operation**

  Add a queen in any square

- **Final State**

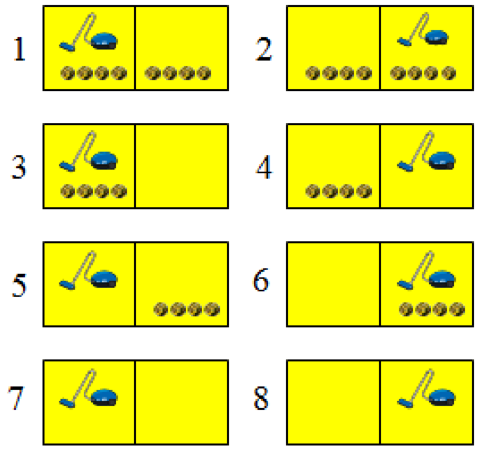


Real World Problems

The Real world problem is related to people actually care about. Such problems are likely not to have a multiple formulations. Route finding, a real world problem is defined in terms of locations and transitions along links between them. Examples: VLSI layout, Robot Navigation

Q. Vacuum World Problem

- **The world has only two *locations***
- **Each location may or may not contain *dirt***
- **The agent may be in one location or the other**
- **8 possible *world states***
- **Three possible actions: *Left, Right, Suck***
- ***Goal*: clean up all the dirt**



Here,

- **Initial States**:

  One of the 8 states

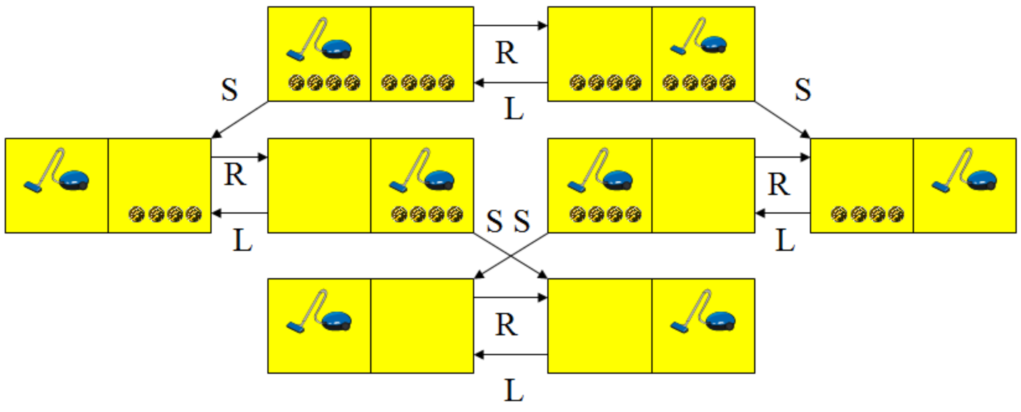
- **Operators:**

  Move left, Move right, Suck

- **Final State**:

  No dirt left in any square

- **Solution**



Well Defined Problems

A problem when defined with its components is called well defined problem. The actions and rules should be defined in as general way as possible.

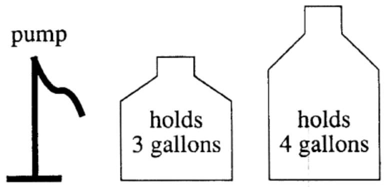
Q.A Water Jug Problem

- **You have a 4-gallon and a 3-gallon water jug**
- **You have a faucet with an unlimited amount of water**
- **You need to get exactly 2 gallons in 4-gallon jug**

Here,

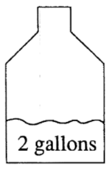
- **Initial State**

  Start state**: (0, 0)**



- **Operators**
  - Fill 3-gallon from faucet, fill 4-gallon from faucet
  - Fill 3-gallon from 4-gallon , fill 4-gallon from 3-gallon
  - Empty 3-gallon into 4-gallon, empty 4-gallon into 3-gallon
  - Dump 3-gallon down drain, dump 4-gallon down drain
- **Final State**

  Goal state **(0, 2)**



- **Solution**

Gallons in the 3-Gallon Jug	**Gallons in the 4-Gallon Jug**
3	3
2	4
2	0
0	2

Q. You are given two jars of a six gallons and eight gallons capacity. There is no marking on the jars. There is a water tap, which can be used to fill jar. Your goal is to have exactly four gallons of water in eight gallons jar without taking any other jar or measuring device. Model the above problem as an AI production system and draw the search tree.

Q. River Crossing Problem

A farmer has a Goat, Wolf, and a Cabbage on the west side of a river. He wants to get all of his animals and Cabbage across the river on the east side. The farmer has raw boat but he only has

enough room for himself and one another thing. The Wolf will eat the Goat if they are left together alone. The Goat will eat Cabbage if they are left together alone. How can the farmer get everything on the east side?

Here,

- **Initial State**

  West {Farmer, Cabbage, Wolf, Goat}&East {ϕ}
- **Operators**

  - Move farmer and one of the Cabbage, Wolf, and Goat from West to East and vice-versa.
- **Final State**

  West {ϕ}   &   East{Farmer, Cabbage, Wolf, Goat}
- **Solution**

S. N	**West**	**River**		**East**
1.	(Farmer, Cabbage, Wolf, Goat)			(ϕ)
2.	(Wolf, Cabbage)	(Farmer, Goat)		(Farmer, Goat)
3.	(Wolf, Cabbage, Farmer)	(Farmer)		(Goat)
4.	(Wolf)	(Farmer, Cabbage)		(Farmer, Cabbage, Goat)
5.	(Wolf, Goat, Farmer)	(Farmer, Goat)		(Cabbage)
6.	(Goat)	(Wolf, Farmer)		(Wolf, Cabbage, Farmer)
7.	(Farmer, Goat)	(Farmer)		(Wolf, Cabbage)
8.	(ϕ)	(Farmer, Goat)		(Farmer, Cabbage, Wolf, Goat)

Constraint Satisfaction Problem (or Crypto Arithmetic Problem)

In artificial intelligence, constraint satisfaction is the process of finding a solution to set of variables V_i (V_1, V_2, …, V_n) and a set of constraint C_i (C_1, C_2, …, C_m), where each variable has a domain of allowed values. There is no any specified rule to define the procedure to solve the CSP; the technique depends upon the kind of constraints being considered. The main features of CSP are:

- A CSP is a high level description of a problem.
- A model for the problem is represented by a set of variables and their domain.
- The problem is stated as constraints specifying the relation between the variables.
- The constraints only specify the relationship without specifying a computational procedure to enforce that relationship.
- The computer has to find the solution of the specified problem.

The CSP is a two-step process: first the constraints are discovered and propagated as far as possible throughout the system then if there is still not found solution, search began. A guess about something is made and added as new constraints. The propagation then occurs with new constraints and forth. The main benefits of CSP are listed as:

- Standard representation pattern

- Generic goal and successor functions
- Generic heuristics (no domain specific expertise).

Problem 1

$$\begin{array}{cccc} & A & B & C \\ \hline + & D & E & F \\ \hline & G & H & I \end{array}$$

Solution:

Tree Searching Rules:

1. $A \neq B \neq C \neq D \neq E \neq F \neq G \neq H \neq I$
2. $C + F = I$

   $C + F = 10 + I$ (I as carry)
3. $B + E = H$

   $B + E = 10 + H$

   $B + E + 1 = H$

   $B + E + 1 = 10 + H$ (H as carry)
4. $A + D = G$

   $A + D + 1 = G$

Now,

Step 1:

Domain of C = {1, 2, 3, 4, 5, 6, 7, 8, 9}

Domain of F = {1, 2, 3, 4, 5, 6, 7, 8, 9}

So, Domain of I = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Select C = 4 & F = 9 Then I = 3 (Carry = 1)

So,

$$\begin{array}{cccc} & A & B & 4 \\ \hline + & D & E & 9 \\ \hline & G & H & 3 \end{array}$$

Step 2:

Domain of B = {0, 1, 2, 5, 6, 7, 8}

Domain of E = {0, 1, 2, 5, 6, 7, 8}

So, Domain of H = {0, 1, 2, 5, 6, 7, 8}

Select B = 2 & E = 8 Then H = 10+1-previous carry = 1 + (Carry = 1)

So,

$$\begin{array}{cccc} & A & 2 & 4 \\ \hline + & D & 8 & 9 \\ \hline & G & 1 & 3 \end{array}$$

Step 3:

Domain of A = {0, 5, 6, 7}

Domain of D = {0, 5, 6, 7}

So, Domain of G = {5, 6}

Select A = 0 & D = 5 Then G = 6 (with addition of Carry)

So,

Hence, the required solutions are:

A = 0, B = 2, C = 4, D = 5, E = 8, F = 9, G = 6, H = 1, I = 3.

- **Problem 3**

$$\begin{array}{ccccc} & S & E & N & D \\ \hline + & M & O & R & E \\ \hline M & O & N & E & Y \end{array}$$

Solution:

Step 1:

Domain of M = {1}

Select M = 1

So we have,

$$\begin{array}{ccccc} & S & E & N & D \\ \hline + & 1 & O & R & E \\ \hline 1 & O & N & E & Y \end{array}$$

Step 2:

Domain of S = {9}

Select S = 9 Then O = 0 (Carry = 1)

So,

$$\begin{array}{ccccc} & 9 & E & N & D \\ \hline + & 1 & 0 & R & E \\ \hline 1 & 0 & N & E & Y \end{array}$$

Step 3:

Domain of D = {2, 3, 4, 5, 6, 7, 8}

Domain of E = {2, 3, 4, 5, 6, 7, 8}

So, Domain of Y = {2, 3, 4, 5}

Select D = 7 & E = 5 Then Y = 2 (Carry =1)

So,

$$\begin{array}{ccccc} & 9 & 5 & N & 7 \\ \hline + & 1 & 0 & R & 5 \\ \hline 1 & 0 & N & 5 & 2 \end{array}$$

Step 4:

Domain of N = {3, 4, 6, 8}

Domain of R = {3, 4, 6, 8}

So, Domain of E will be 5 (with carry = 1) when we **select** N = 6 & R = 8.

Now we have,

$$\begin{array}{ccccc} & 9 & 5 & 6 & 7 \\ \hline + & 1 & 0 & 8 & 5 \\ \hline 1 & 0 & 6 & 5 & 2 \end{array}$$

Hence, the required solutions are:

S = 9, E = 5, N = 6, D = 7, M = 1, O = 0, R = 8, Y = 2.

Production System

Knowledge representation formalism consists of collections of condition-action rules. A system that uses this rule of knowledge representation is called a production system.

A production system consists of rules and factors. Knowledge is encoded in a declarative from which comprises of a set of rules of the form

Situation ------------ Action

→ SITUATION that implies ACTION.

Example:-

IF the initial state is a goal state THEN quit.

The major components of an AI production system are

i. A global database

ii. A set of production rules and

iii. A control system

- The goal database is the central data structure used by an AI production system
- The production rules operate on the global database. Each rule has a precondition that is either satisfied or not by the database. If the precondition is satisfied, the rule can be applied. Application of the rule changes the database.
- The control system chooses which applicable rule should be applied and ceases computation when a termination condition on the database is satisfied.

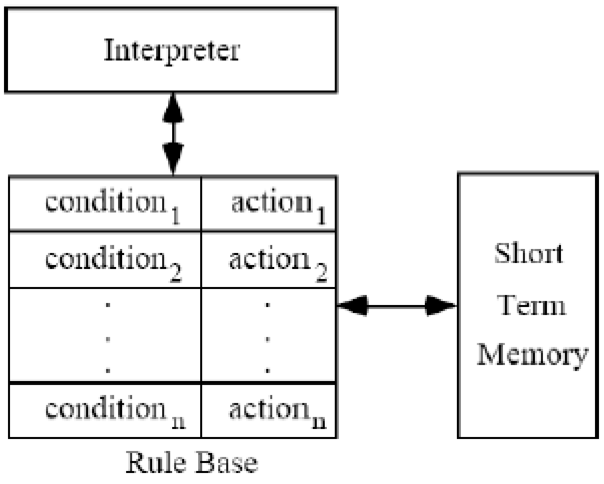
Advantages:

- Production systems provide an excellent tool for structuring AI programs.
- Production Systems are highly modular because the individual rules can be added, removed or modified independently.
- The production rules are expressed in a natural form, so the statements contained in the knowledge base should the recording of an expert thinking out loud.

Disadvantages:

- One important disadvantage is the fact that it may be very difficult to analyze the flow of control within a production system because the individual rules don’t call each other.

Architecture of Production System



Short Term Memory:

- Contains the description of the current state.

Set of Production Rules:

- Set of condition-action pairs and defines a single chunk of problem solving knowledge.

Interpreter:

- A mechanism to examine the short term memory and to determine which rules to fire