classmate
Date
Page

ARTIFICIAL INTELLIGENCE

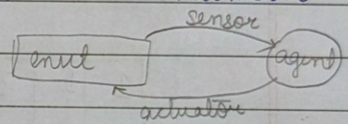
Man-made — The capacity of learning, reasoning, understanding, act like human being, thinking like human being.

A problem

→ Definitions :-

→ Artificial Intelligence is an area of study in CSE, other engg or basic science through which an AI enabled device or machine can be developed, which can behave like a human being.

→ It can be defined in terms of agent. An agent is anything that can be viewed or receiving information from the environment through sensors & act like a human being through actuators.




→ Application areas of AI :-
Nowadays, AI can be applied in any area whereever a computational problem is to be solved.

→ Computational problem / Problem :-
A / computational problem or a problem is defined through its elements & their relationships.
Problem space : It contains all possible elements of a problem within a definite space or area.
State : A state is an instance of a problem.
State space : All possible states for a problem. (OR) It contains all possible relevant objects which are associated with a problem.

Each state space is associated with two states 1. Initial state & 2. Final state / Goal state.

Maximum AI problems can be viewed through a searching problem.

Eg: 8 puzzle game.

For this game

- State space : Configuration of 8 tiles on the board
- Initial state : any configuration.
- Goal state : tiles in specific order
- Action : blank moves.

Condition : the move is within the board.

Transformation : blank moves left, right, Up, Dn.

- Solution : optimal sequence of operators

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | |

Solution

←

| | | |
|---|---|---|
| 3 | 2 | 5 |
| 4 | 1 | 6 |
| | 8 | 7 |

Initial state

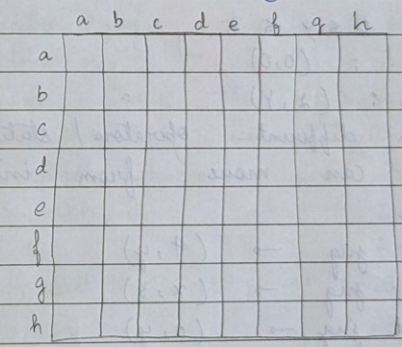
A × B × C × D

1. ((A × B) × (C × D))
2. (A × ((B × C) × D))
3. (A × (B × (C × D)))
4. (((A × B) × C) × D)
5. ((A × (B × C)) × D)

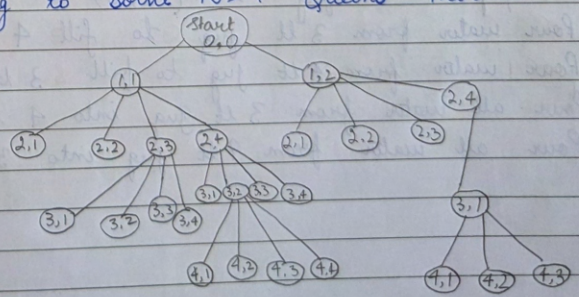
N- Queen Problem :

A game of n-queens puzzle n = 8

- State space : Configurations n = 8 queens on the board with only one queen per row & columns
- Initial state : Configuration without queens on the board.
- Goal state : Configuration with n = 8 queens such that that no queen attacks any other
- Operations or actions : Place a queen on the board.
  - Condition : the new queen is not attacked by any other already placed.
- Transformations : Place a new queen in a particular square of the board.
- Solution : one solution (cost is not considered).




Backtracking to solve N = 4 Queens Problem :



## WATER JUG PROBLEM:

In case of this problem we are given two jugs a 4 litre jug & a 3 litre jug

- Unlimited water is available
- Neither of the jugs has any measuring marking on it.

We have to find out how can we get exactly 2 litre of water using 4 litre jug.

**State representation of the Problem:**

We can represent the state of the problem using ordered pair $(x, y)$ where $x$ represent amount of water in 4 litre jug & $y$ represents amount of water in 3 litre jug

i.e. $0 \leq x \leq 4$

$0 \leq y \leq 3$.

- Initial state : $(0,0)$
- Final state : $(2, y)$

There are different operators / states through which we can move from initial state to final state.

1. Fill 4 lt jug → $(4, y)$
2. Fill 3 lt jug → $(x, 3)$
3. Empty 4 lt jug → $(0, y)$
4. Empty 3 lt jug → $(x, 0)$
5. Pour water from 3 lt jug to fill 4 lt jug → $(4, y-(4-x))$
6. Pour water from 4 lt jug to fill 3 lt jug → $(x-(3-y), 3)$
7. Pour all water from 3 lt jug into 4 lt jug → $(x+y, 0)$
8. Pour all water from 4 lt jug into 3 lt jug → $(0, x+y)$

| | 4 lt. jug | 4, 3 | 3 litre jug | State |
|---|---|---|---|---|
| 1. | 0 | (0, 0) | 0 | |
| 2. | 4 | (4, 0) | 0 | Rule 1 |
| 3. | 1 | (1, 3) | 3 | Rule 6 |
| 4. | 1 | (1, 0) | 0 | Rule 4 |
| 5. | 0 | (0, 1) | 1 | Rule 3 |
| 6. | 4 | (4, 1) | 1 | Rule 1 |
| 7. | 2 | (2, 3) | 3 | Rule 6 |
| 8. | 2 | (2, 0) | 0 | Rule 4 |

Example: Let there are 3 jugs available - 8 lt, 5 lt & 3 lt. Get 2 lt of water using 5 lt jug.

$(x^8, y^5, z^3)$

1. Fill 5 litre jug → $(x, 5, z)$
2. Fill 8 litre jug → $(8, 5, z)$
3. Fill 3 litre jug → $(8, 5, 3)$ ✓
3. Empty 8 → $(0, 5, 3)$
6. Fill 8 with 5 → $(5, 0, 3)$
7. Fill 5 with 3 → $(5, 3, 0)$
6. Fill 3 with 8 → $(2, 3, 3)$

Empty 5 → $(2, 0, 3)$

Fill 5 with 8 → $(0, 2, 3)$

| | 8 lt | 5 lt | 3 lt | (x, y, z) | state |
|---|---|---|---|---|---|
| 1. | 0 | 5 | 0 | (0, 5, 0) | fill 5 |
| 2. | 0 | 2 | 3 | (0, 2, 3) | pour 5 to 3 |
| 3. | 0 | 2 | 0 | (0, 2, 0) | empty 3 |

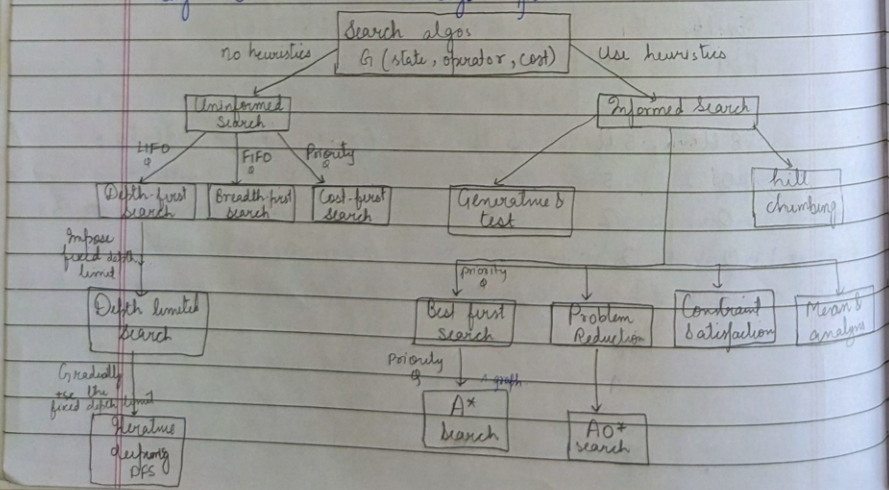
→ DIFFERENT SEARCHING STRATEGIES / HIERARCHICAL REPRESENTATION OF SEARCHING ALGORITHMS.

Basically there are two types of searching algorithm

1. Uninformed searching algorithm.
2. Informed searching algorithm.

1. Uninformed searching algo:
Also known as blind search, exhaustive search or iterative searching algorithms.
They use exhaustive / iterative strategies.
These type of searching algorithms are not optimal (always) as it needs to traverse all possible states from initial state to final.

2. Informed searching algo:
Also called heuristic or intelligent search, uses information for the problem to guide the search, usually guesses the distance to a goal state & therefore efficient, but the search may not be always possible.




DS, AI, ML, DL → common 1. Data 2. Maths

Heuristics Search Technique

For complex problems, the traditional algorithms (DFS, BFS, Floyd Warshal) are unable to find the solution within a practical time or within a space limit. So to solve this problem, different heuristic algos are used.

The heuristic algos are basically based on heuristic function which is represented as $h(n)$ where $n$ is the instance of the problem P. & $h(n)$ is equal to heuristic value at instance $n$.

Heuristic search is a weak technique but can be effectively applied if it applied correctly.

That means it requires domain specific info or knowledge.

When we estimate the value for $h(n)$ it may be over estimate or under estimate.

Eg;

'Initial state'

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 7 | 8 | 4 |
| 6 | | 5 |

'Goal state'

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 8 | | 4 |
| 7 | 6 | 5 |

left | right | up

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 7 | 8 | 4 |
| | 6 | 5 |

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 7 | 8 | 4 |
| 6 | 5 | |

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 7 | | 4 |
| 6 | 8 | 5 |

if h(n) then h(n) = 6

| Approach | left | right | up |
|---|---|---|---|
| 1. Count the correct position | 6 | 4 | 5 |
| 2. Count the incorrect position | 2 | 4 | 3 |
| 3. Count how far away | 2 | 4 | 4 |

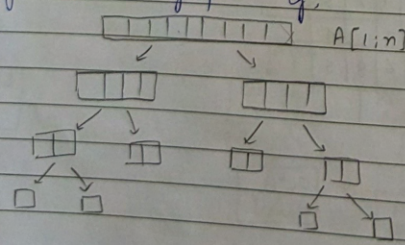
There are different metrics to define heuristic functions.

1) Euclidean distance
2) Geo-disc distance
3) Cosine distance
4) Manhattan distance
5) Hamming distance

AND Graph & AND OR Graph:
Based upon the nature of the problem, a problem can be represented using 2 different graphs.

1. AND graph 2. AND OR graph

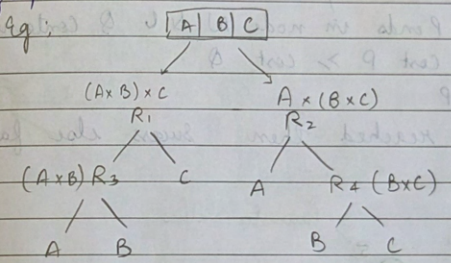
(i) AND Graph: If the solution of the problem depends upon all the paths from the source node, then that problem can be represented by using AND graph. Eg:




Eg: Merge sort & quick sort - the subproblems can be represented by an AND graph to find out solution of original problem.

(ii) AND-OR Graph: If a problem can be divided into different sub problems & the subproblems are independent interms of solutions, then those types of problems can be represented by AND OR graph.

Eg:




Matrix chain mul. is one of the ex. which can be rep. by using AND OR graph.

A* ALGORITHM : AND graph

Let there be a graph G. associated with n no. of vertices & each vertex has a heuristic value. There is a distance from each pair of vertex & the heuristic value of a node is represented by h(n) & the actual distance bw each pair of the vertex is represented by G(n), then the accumulated cost of a node n is represented by $f(n) = h(n) + g(n)$.

A graph G is taken as input to the algorithm & shortest path from initial state to goal state is the output.

Priority Queue: This is a data structure which is used to keep information about the shortest path from the source node to all other nodes which are considered as one-one paths.

Input - Queue : Path only Containing root

Algorithm

While ( queue not empty && first path not a reaching )

Do

- Remove first path from Queue
- Create paths to all children
- Reject paths with loops
- Add paths & sort queue (by $f$ = Cost + heuristic)
- If queue contains paths : P, Q

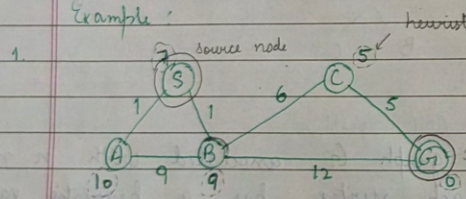
  AND P ends in node N & Q contains node N

  AND cost P >= cost Q

Then remove P

- If goal is reached then Success else failure

Example :




Q = ____ Empty initially

$f(A) = g(A) + h(A) = 1 + 10 = 11$

$f(B) = g(B) + h(B) = 1 + 9 = 10$

Q = SB (10) SA (11)

Q = SA (11) SBA (20) SBC (12) SBG (13)

both end with A not necessary to include

Q = SA (11) SBC (12) SBG (13)

SBA → not include as B at end, here in middle

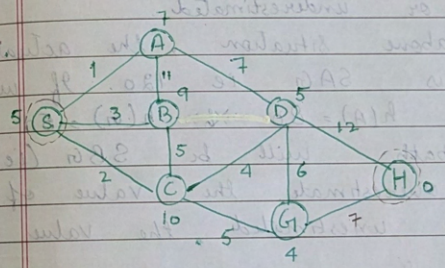
Q = SBC (12) SBG (13) SAB (19) P > Q, 19 > 12 → hence remove

Q = SBC (12) SBG (13)

Q = SBCG (12) SBG (13) (both end with G)

S - B - C - G is the path with shortest distance

2.



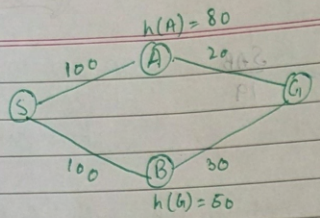

Q = SA (8) SB (12) SC (12)

Q = SB (12) SC (12) SAD (13) SAB (21)

Q = SC (12) SAD (13) SBC (18) SBA (21)

Q = SCG (11) SCD (13) SAD (16) SCB (18) SBA (21)

Q = SCGH (14) > SCDH (18)



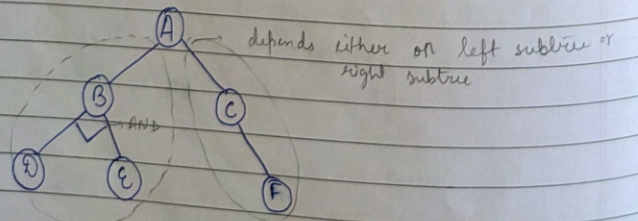

A* algorithm is admissable or not depends upon the estimated or heuristic value. Based upon the estimated value the result may be overestimated or underestimated.

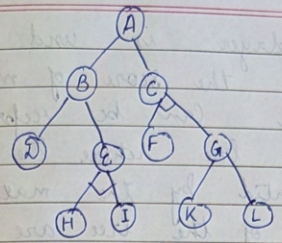
Consider the above situation, the actual shortest path is SAG ie 120. If we estimate value of $h(A) = 80$ & $h(G) = 50$ the shortest path will be SBG (ie 130) as we overestimate the value of $h(A) = 80$. And unestimated the value of $h(B) = 50$.

But in reverse case, we can get actual answer.

## AO* Algorithm :

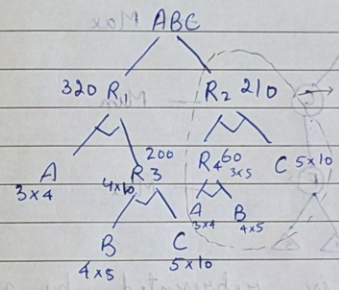
AO* algorithm is applicable for AND OR graph. AND OR graph is used to represent the different subproblems of a problem P where the solution of the problem P may depend upon any one of the subproblem which can be represented by using an AND OR tree also.



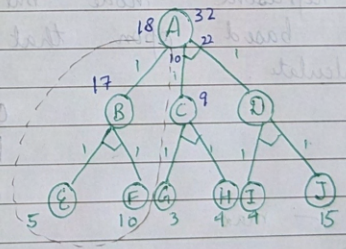




→ Example: Let there be 3 matrices

$A_{3\times4}$, $B_{4\times5}$ & $C_{5\times10}$




Q

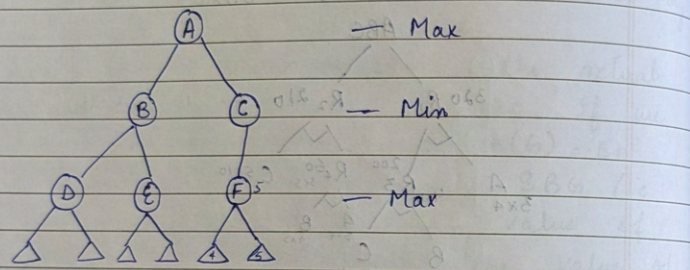



## Game Tree

A game tree is a representation of all possible paths from initial state to goal state.

Here the game tree can be represented by using a minmax tree which is like two players one is a max player & other is min player. The max player is used to maximise or increment the score towards to find out soln

of problem & min player is used to minimise or decrement the score of max player. Then the whole problem can be represented by using a min max tree where the root node of the tree is represented by the max player & rest of the levels of the tree are represented alternatively.



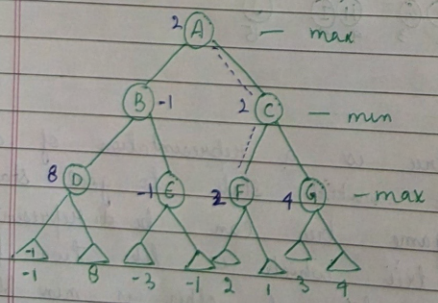

The last level is represented by a triangle It doesnot represent node but it represents the cost & based upon that value we have to calculate.

$O(b^d)$

b → branch factor

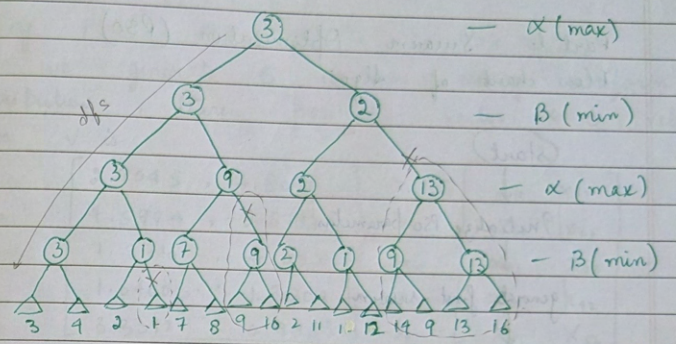
d → depth.

Eg




Path = A - C - F

## α β Prunning Algorithm:




Step 1: Do the DFS until to reach the first leaf node. Here α is the highest value choice found at any point of the path for the max player & β is the lowest value found at any point of the path for the min player. Then pass the current value of α & β to the child node during the search.

Step 2: Update the value of α & β through max & min player respectively.

Step 3: Prune the remaining branches at a node when $\alpha \geq \beta$.

Step 4: Cut off below the max node where α values becomes $\alpha \geq \beta$ value of each successor.

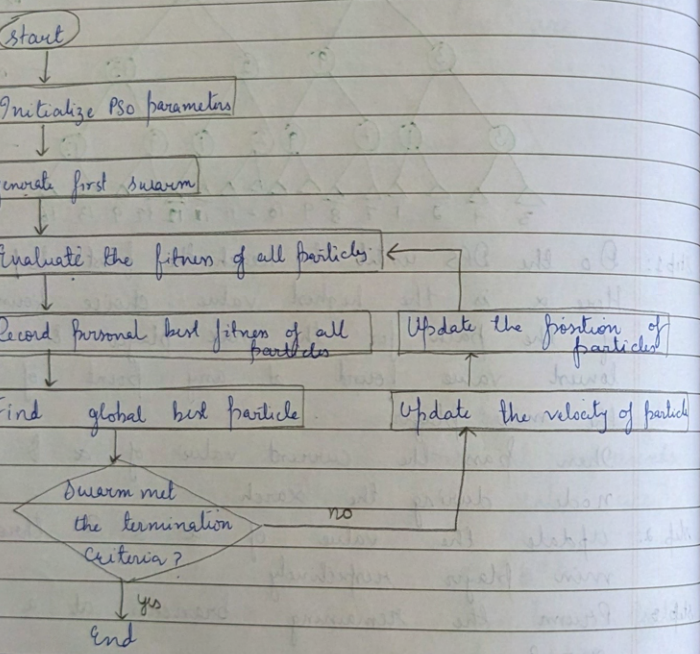
Step 5: Cut off the min node whose β value is $\leq \alpha$ value of its successor.

The Complexity of the above algo will be $O(b^{d/2})$ where b is the branch factor & d is the depth of the tree.

Constant Stisfaction Problem: X

Particle Swarm Optimization (PSO)

Flow chart of Algo:




Ex To optimize the func $f(x) = x_1^2 + x_2^2$.
Here $-5 \le x_1, x_2 \le 5$
Using PSO minimize the function $f(x)$

Sol Let we generate 5 swarm using uniform distribution, their position vector X & velocity vector V is

$$X = \begin{bmatrix} 2.7045, & 4.8030 \\ 4.5974, & 2.8793 \\ 1.8710, & 4.0528 \\ 1.6400, & 1.3202 \\ 3.3392, & 0.9963 \end{bmatrix} \begin{matrix} -x_1 \\ -x_2 \\ -x_3 \\ -x_4 \\ -x_5 \end{matrix} \quad \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \\ x_{41} & x_{42} \\ x_{51} & x_{52} \end{bmatrix}$$

$$V = \begin{bmatrix} 0.4752, & 0.6987 \\ 0.4141, & 0.4020 \\ 0.7797, & 0.9433 \\ 0.6183, & 0.4749 \\ 0.2530, & 0.9398 \end{bmatrix} \begin{matrix} -v_1 \\ -v_2 \\ -v_3 \\ -v_4 \\ -v_5 \end{matrix} \quad \begin{bmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \\ v_{31} & v_{32} \\ v_{41} & v_{42} \\ v_{51} & v_{52} \end{bmatrix}$$

The velocity vector generated uniformly in range [0,1].

By using the initial fitness vector fitness value of each swarm particle is,

for $x_1 = f(x_1) = (2.7045)^2 + (4.8030)^2 = 30.3831$
$x_2 = f(x_2) = (4.5974)^2 + (2.8793)^2 = 29.4265$
$x_3 = f(x_3) = (1.8710)^2 + (4.0528)^2 = 19.9258$
$x_4 = f(x_4) = (1.6403)^2 + (1.3202)^2 = 4.4325$
$x_5 = f(x_5) = (3.3392)^2 + (0.9963)^2 = 12.1429$

Here the minimum value is 4.4325, ie for the swarm $x_4$, which is the best soln of ($0^{th}$ iteration) at this time.

So 4.4325 is the gbest value of particle $x_4$.

As this is the 0th iteration, so each particle current position is also the pbest position.

$$v_i^{t+1} = v_i^t + c_1 r_1^t (pbest_i^t - p_i^t) + c_2 r_2^t (gbest^t - p_i^t)$$

$v_i^t$ → inertia; $c_1 r_1^t (pbest_i^t - p_i^t)$ → personal influence; $c_2 r_2^t (gbest^t - p_i^t)$ → Social influence

The position vector will be.

$$p_i^{t+1} = p_i^t + v_i^{t+1}$$

Here $c_1 = c_2 = 2$ $(0 \le c_1, c_2 \le 2)$

$r_1 = 0.34$

$r_2 = 0.86$ $\{0 \le r_1, r_2 \le 1\}$

iter 1 (velocity of $v_i'$)

$$v_1' = v_{11}^0 + c_1 r_1 (pbest_1^0 - x_{11}^0) + c_2 r_2 (gbest_1^0 - x_{11}^0)$$

$$= 0.475 + 2 \times 0.34 (2.7045 - 2.7045) + 2 * 0.86 (1.6400 - 2.7045)$$

$$\approx -1.35574$$

Position of $x_1'$

$$\Rightarrow x_{11}' = x_{11}^0 + v_{11}'$$

$$= 2.7045 + (-1.35574)$$

$$= 1.34876.$$

Now, we have to check whether it is within (-5 to 5) So, it is acceptable.

for 2nd component ($x_{12}$)

$$v_{12}' = v_{12}^0 + c_1 r_1 (pbest_{12}^0 - x_{12}^0) + c_2 r_2 (gbest_{12} -$$

Continue Assignment

## Constraint Satisfaction problem (CSP):

The problem which occur in case of BFS, DFS & traditional backtracking technique can be overcome by using the principle of Constant Satisfaction prob. The CSP can be defined with three tuples / Components $\{V, D, C\}$ where

$V$ is the set of variables $\{v_1, v_2, v_3 \dots v_n\}$

$D$ = set of domains = $\{d_1, d_2, d_3 \dots d_m\}$

$C$ = Set of Constraint = $\{C_1, C_2 \dots$ that specifies the allowable combination of variables

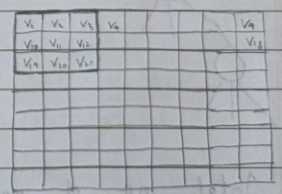
$C_i$ = {scope, relation}

Sudoko problem is one of the problem which can be solved by using CSP. Here a 9 x 9 matrix is considered

$V = \{v_1, v_2, v_3 \dots v_{81}\}$

$D = \{1, 2, 3, 4 \dots 9\}$

$C = \{C_1 = \{$

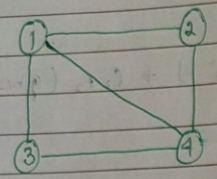


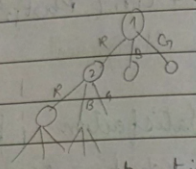
## Graph Colouring Problem :-

Graph Coloring Problem can be solved by using CSP. Here the set of variables are the set of vertex of the graph G & the domain of this problem is the no of colors we want to assign to the vertex ($D$). Here the constraint is

no two adjacency vertex has the same color.

Ex:




$3^{n+1}-1$

$3^0$

$3^1$

$3^2$

$3^3$

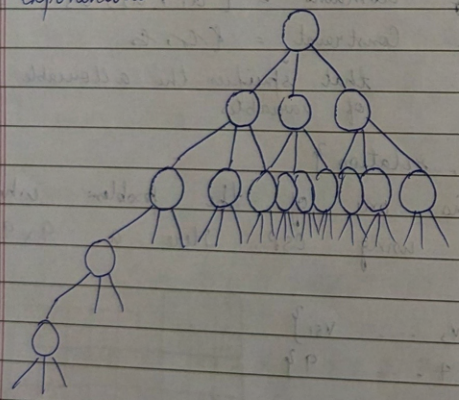
exp time complexity.

V = {1, 2, 3, 4}

D = {R, G, B}

C = { 1 ≠ 2 (color of), 1 ≠ 4, 1 ≠ 3, 2 ≠ 4, 3 ≠ 4 }

To explore the above problem using iterative method the complexity of the problem is exponential.



$3^0$

$3^1$

$3^2$

$3^3$

$3^4$

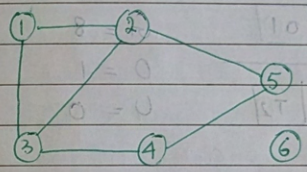
Total no. of nodes (N) = $3^0 + 3^1 + 3^2 + 3^3 + 3^4 + \ldots$

= $C^{n+1} - 1$

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Initial State | RGB | RGB | RGB | RGB |
| 1 = R | R | GB | GB | GB |
| 2 = G | R | G | GB | B |
| 3 = B | R | G | B | B |
| 3 = G | R | G | G | B |

→ Here the condition violates as blue is already assigned to 4 & we can't assign to 3rd vertex. So we have to use backtracking.

Ex




V = {1, 2, 3, 4, 5, 6}

D = {R, G, B}

C = { 1 ≠ 2, 1 ≠ 3, 2 ≠ 3, 2 ≠ 5, 3 ≠ 4, 4 ≠ 5 }

## Crypto Arithmetic :-

In this problem, the variables are A, B, C ... Z or a, b, c .... z & domains are {0, 1, 2, 3 ... 9} & the constant C = {no two letters have the same value, sum of digit must be as per the problem & their should any one of the carry (not more than 1)}

Ex 1.

```
  TO        T(2)   O(1)
  GO
  OUT   +   G(8)   O(1)
 ------------------------
  O(1)      U(0)   T(2)
```

T = 2
G = 8
O = 1
U = 0

$C_0$ $C_3$ $C_2$ $C_1$

2.

```
  SEND      S=9   E=5   N=6   D=7
+ MORE      M=1   O=0   R=8   E=5
 ------------------------------
  MONEY     M=1   O=0   N=6   E=5   Y=2
```

8/7/6/5/4/3/2

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

Assignment : Soln of 8-queen problem by using constraint satisfaction problem.

## GREY WOLF OPTIMIZATION

Algo:

1. Initialize the parameters : Population size, maximum iteration
2. Find the best ($X_\alpha$), 2nd best ($X_\beta$) & 3rd best ($X_\delta$) positions.
3. for Iteration = 1
   - Compute $a = 2\left(1 - \frac{\text{iteration}}{\text{max itr}}\right)$
   - Compute $X_1, X_2, X_3$
   - Compute $X_{new} = \frac{X_1 + X_2 + X_3}{3}$
   - Check the bounds
   - Perform Greedy Selection ie $f(X_{new})$ is better or not. If yes, update the solution.
4. Set iteration = iteration + 1 & go on to step 2.

Example $f(x) = 2x_1^2 + x_2 + 3x_1x_2$

Minimize this function using GWO

Range (-10 to +10)

Generate 5 different position vector (of wolves) By implementing (as by convergence) for copy 2

In assignment write :

1. Intro to algo
2. Mathematical interpretation
3. Pseudo code or & flow chart of Algo
4. Example (Soln upto 2 iterations)
5. Code implementation & Output (in Python).

## Fuzzy Logic :

In real world many problems exist based on fuzzy that means the elements or the attributes which are associated with the problem are not clear. OR we can say the values are in ambiguous ( different person can interpret in different directions).
In this case the elements of the problem are not expressed or represented through classical set.
And also the human thinking & reasoning are changed based upon different situations & the nature will be in fuzzy.
So to express the human thinking & reasoning cannot be possible through classical set theory, which can be expressed through fuzzy set theory.

- Classical set theory allows the membership of the elements in a set in binary (0,1) where as the fuzzy set theory allow the membership value of an element in between 0 to 1. And the membership value of the element is to be assigned through membership function.

- Words like young, tall, good or high are said to be fuzzy & these are known as fuzzy words.
There is no single quantitative value which defines the term young. For some people 25 is young & for others 35 is young.
It has no clear boundary. Age 35 has some possibility of being young & usually depends on context.

→ Fuzzy set theory is an extension of classical set theory where elements have degree of membership

Example: Define a fuzzy set S for young students within a section where the age is from 25 to 28.

Let $S_1 = 25$, $S_2 = 28$, $S_3 = 27.5$, $S_4 = 26$

$S_5 = 25.7$

$S_1 > S_4 > S_3 > S_2$

$S = \{(S_1, 0.9)\ (S_2, 0.7)\ (S_3, 0.75)\ (S_4, 0.8)\}$

classical set = $\{S_1, S_2, S_3, S_4\}$

→ A fuzzy set can be represented using $\tilde{A}$ {
The membership value of the elements can be defined through a membership function represented by $\mu_{\tilde{A}}(x)$.

$\tilde{A} = \{(x_1, 0.9), (x_2, 0.8), (x_3, 0.4)\ (x_4, 0.1)\}$

$\mu_{\tilde{A}}(x_1) = 0.9$

$\mu_{\tilde{A}}(x_4) = 0.1$

Here 0.1 represents / indicates the belongingness or membership value of the element $x_4$ in the fuzzy set A.

→ Fuzzy set theory operations
Given X to be the universe of discourse & $\tilde{A}$ & $\tilde{B}$ to be fuzzy sets with

$\mu_A(x)$ & $\mu_B(x)$. Then the fuzzy set operations.

Union = $\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$

Intersection = $\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$

Complement = $\mu_{\bar{A}}(x) = 1 - \mu_A(x)$

Example: $A = \{(x_1, 0.5), (x_2, 0.7), (x_3, 0)\}$
$B = \{(x_1, 0.8), (x_2, 0.2), (x_3, 1)\}$

$A \cup B = \{(x_1, 0.8), (x_2, 0.7), (x_3, 1)\}$
because
$\mu_{A \cup B}(x_1) = \max(\mu_A(x_1), \mu_B(x_1))$
$= \max(0.5, 0.8)$
$= 0.8$

$\mu_{A \cup B}(x_2) = 0.7 \qquad \mu_{A \cup B}(x_3) = 1$

$A \cap B = \{(x_1, 0.5), (x_2, 0.2), (x_3, 0)\}$
because
$\mu_{A \cap B}(x_1) = \min(\mu_A(x_1), \mu_B(x_1))$
$= \min(0.5, 0.8)$
$= 0.5$

Complement $A^c = \{(x_1, 0.5), (x_2, 0.3, (x_3, 1)\}$
because $\mu_A(x_1) = 1 - \mu_A(x_1)$
$= 1 - 0.5$
$= 0.5$