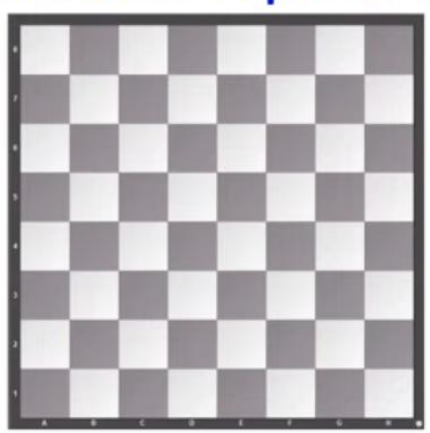
Unit 3-
Local Search and Optimization Problems

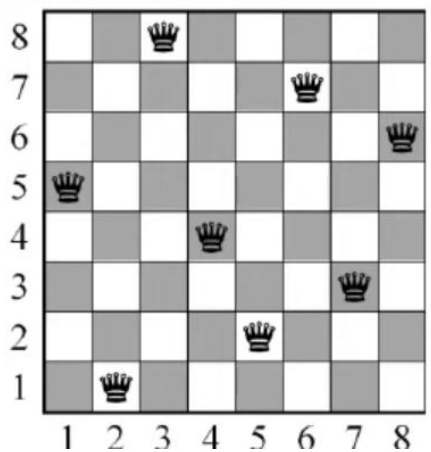
Heuristic Function in AI

- Consider the following 8-puzzle problem where we have a start state and a goal state. Our task is to slide the tiles of the current/start state and place it in an order followed in the goal state.
- There can be four moves either **left, right, up, or down**. There can be several ways to convert the current/start state to the goal state, but, we can use a heuristic function h(n) to solve the problem more efficiently.

Local Search Algorithm

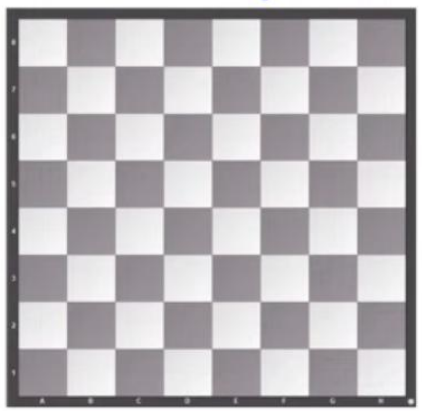
- The Local search algorithm searches only the final state, not the path to get there.
- For example, in the 8-queens problem,
- we care only about finding a valid final configuration of 8 queens (8 queens arranged on chess board, and no queen can attack other queens) and not the path from initial state to final state.
-

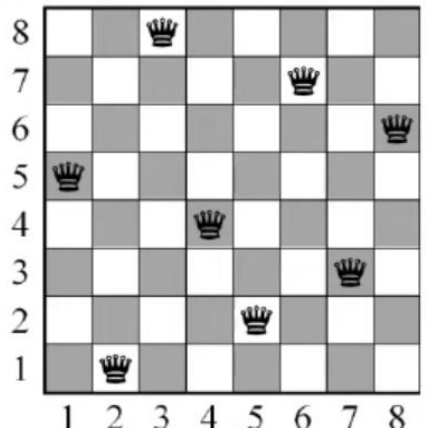




Local Search Algorithm

- The Local search algorithm searches only the final state, not the path to get there.
- For example, in the 8-queens problem,
- we care only about finding a valid final configuration of 8 queens (8 queens arranged on chess board, and no queen can attack other queens) and not the path from initial state to final state.
-



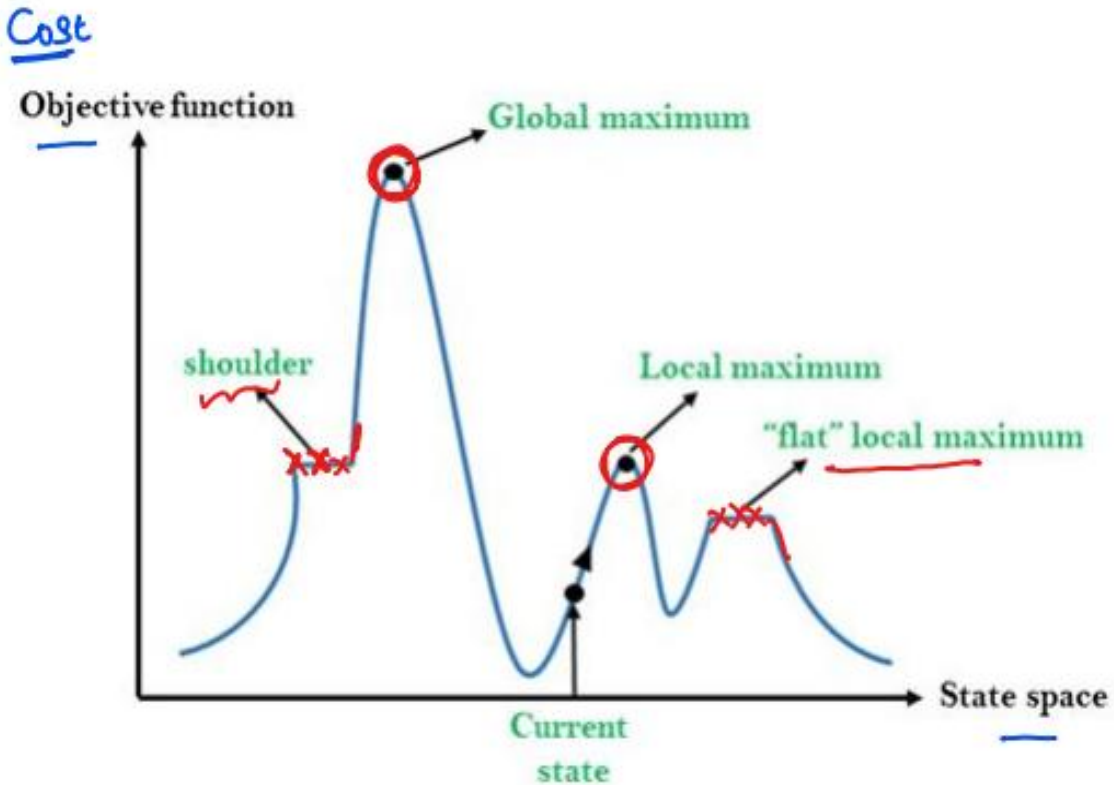


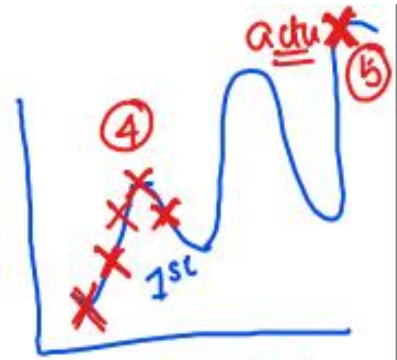
Local Search Algorithms

- **Local search** algorithms operate by searching from a start state to neighboring states,
- without keeping track of the paths, nor the set of states that have been reached.
- They are not systematic—
- they might never explore a portion of the search space where a solution actually resides.
- They searches only **the final state**

Local Search

- The search algorithms that we have seen so far are designed to explore search spaces systematically.
- This systematicity is achieved by keeping one or more paths in memory and by recording which alternatives have been explored at each paint along the path.
- When a goal is found, the path to that goal also constitutes a solution to the problem.
- In many problems, however, the path to the goal is irrelevant. For example, in the 8-queens problem.
- If the path to the goal does not matter, we might consider a different class of algorithms,ones that do not worry about paths at all. Local search algorithms operate using a single current node (rather than multiple paths) and generally move only to neighbors of that node.
- local search algorithms are useful for solving pure optimization problems, in which the aim is to find the best state according to an objective function





downhill

upwards

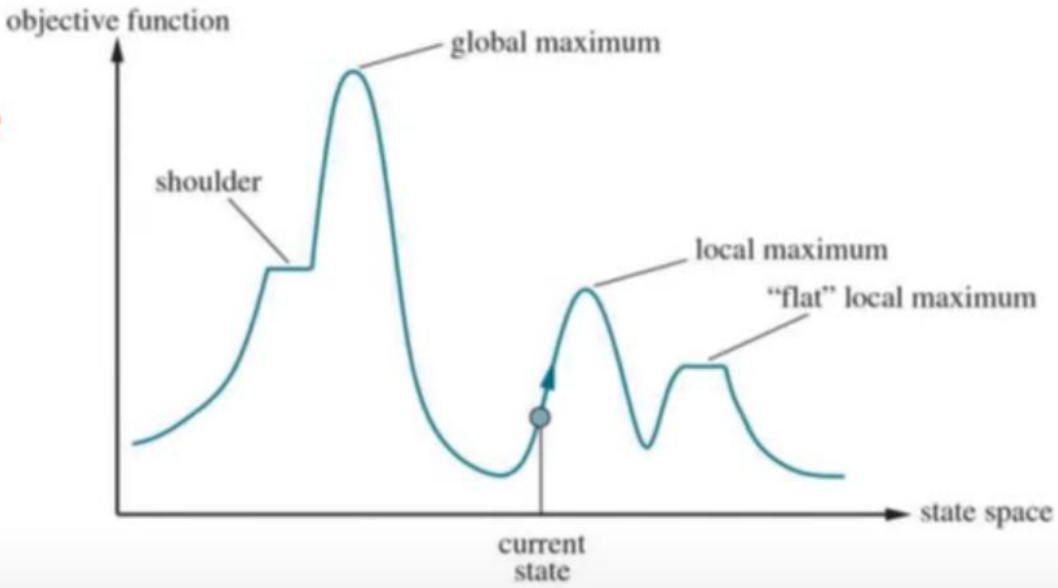
Stop

5 - global max.

4 - local max.

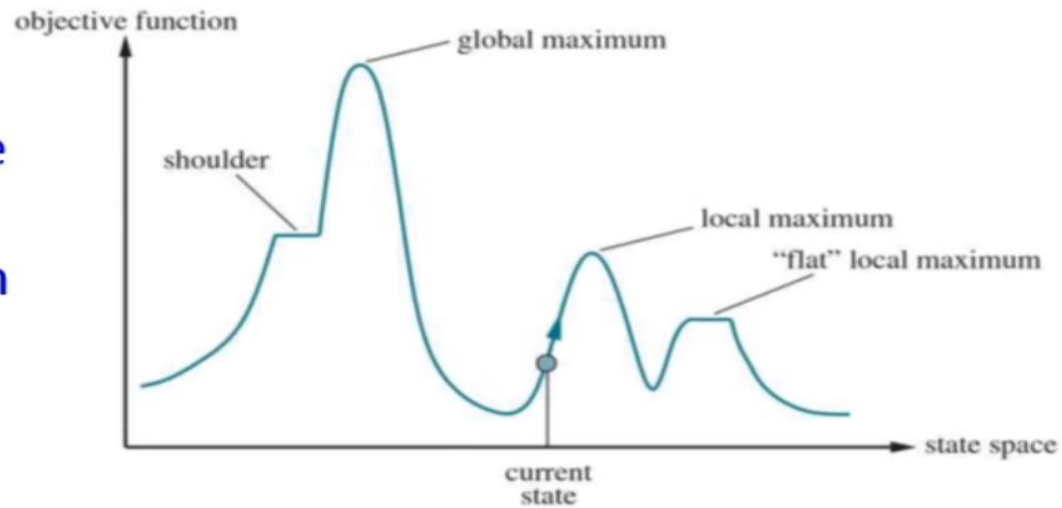
Hill-climbing Search Algorithm

- Hill climbing algorithm is a Heuristic search algorithm which continuously moves in the direction of increasing value to **find the peak of the mountain or best solution to the problem**.
- It keeps track of one current state and on each iteration moves to the neighboring state with highest value—that is, it heads in the direction that provides the **steepest ascent**.



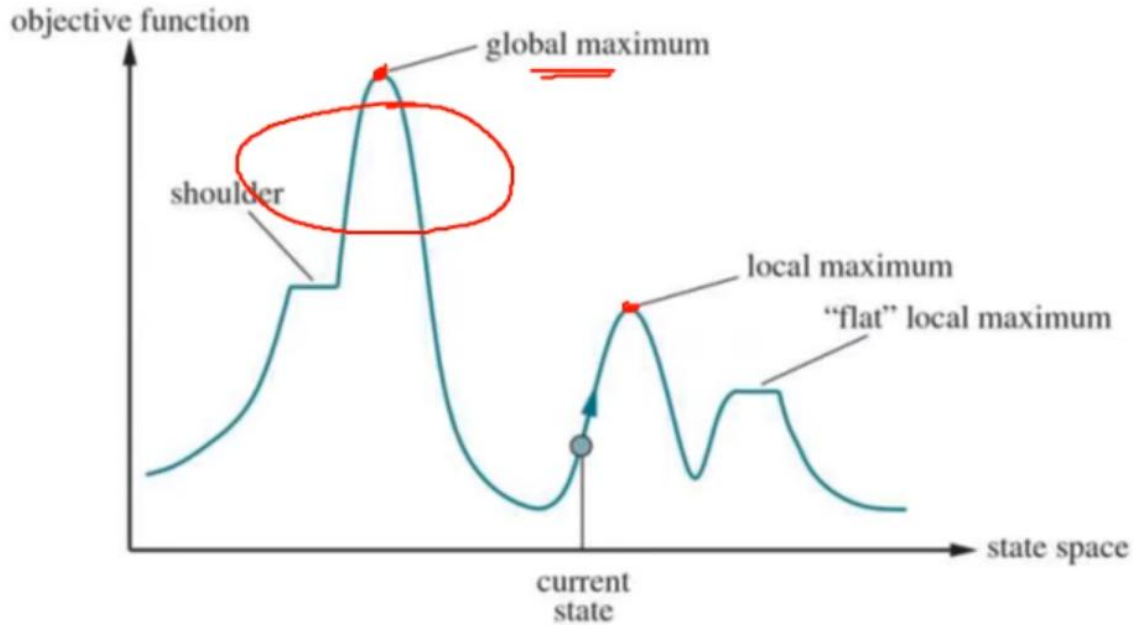
Hill-climbing Search Algorithm

- Different regions in the state space landscape:
- **Local Maximum** is a state which is better than its neighbor states, but there is also another state which is higher than it.
- **Global Maximum** is the best possible state of state space landscape. It has the highest value of objective function.
- **Current state** is a state in a landscape diagram where an agent is currently present.
- **Flat local maximum** is a flat space in the landscape where all the neighbor states of current states have the same value.
- **Shoulder** is a plateau region which has an uphill edge.

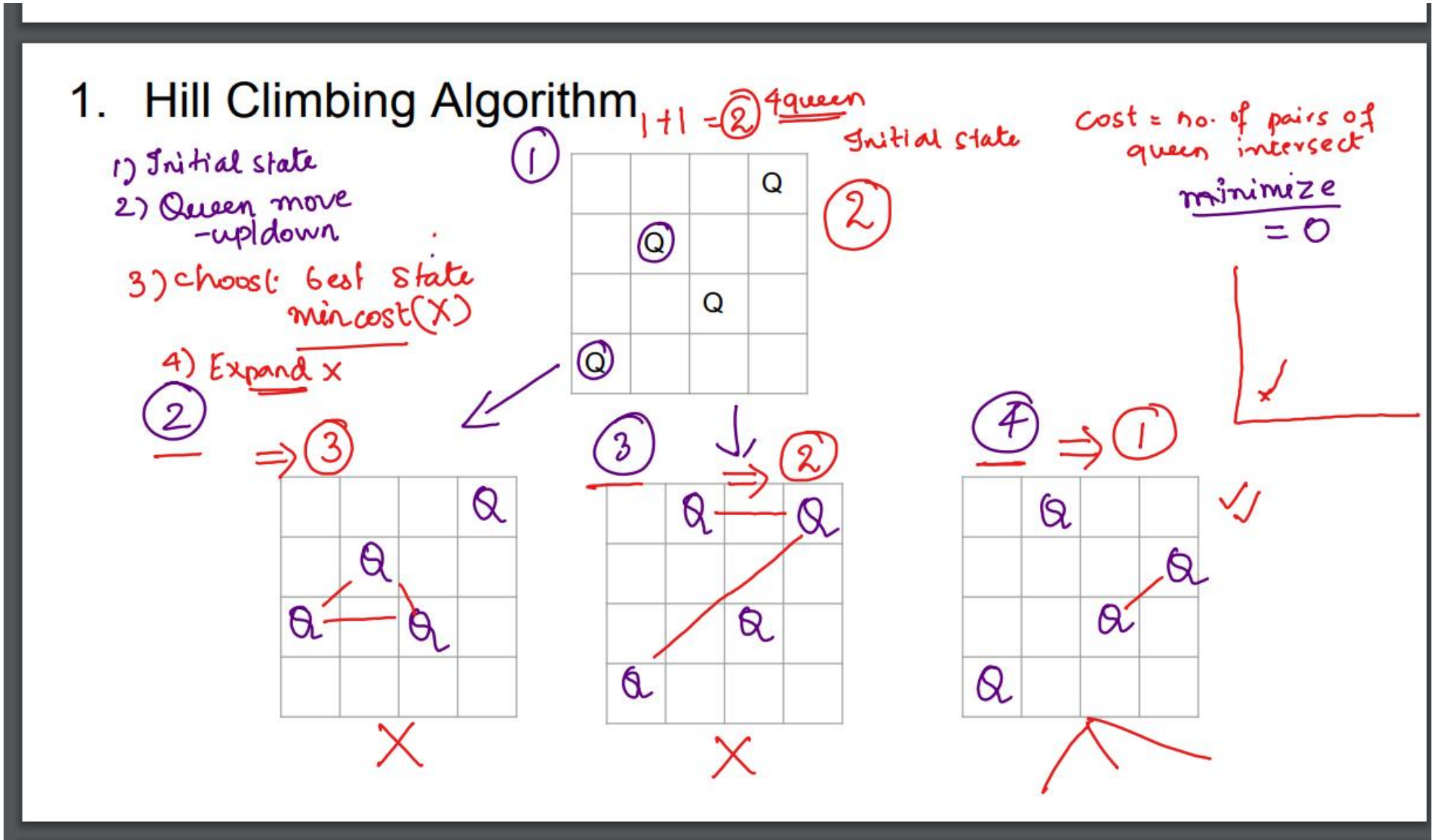


Hill-climbing Search Algorithm...

- In this algorithm, when it reaches a peak value where no neighbor has a higher value, then it terminates.
- It is also called greedy local search as it only searches its good immediate neighbor state and not beyond that.
- Hill Climbing is mostly used when a good heuristic is available.



Hill Climbing- 4 QUEEN PROBLEM



```
function HILL-CLIMBING(problem) returns a state that is a local maximum
  current ← MAKE-NODE(problem.INITIAL-STATE)
  loop do
    neighbor ← a highest-valued successor of current
    if neighbor.VALUE ≤ current.VALUE then return current.STATE
    current ← neighbor
```

Figure 4.2 The hill-climbing search algorithm, which is the most basic local search technique. At each step the current node is replaced by the best neighbor; in this version, that means the neighbor with the highest VALUE, but if a heuristic cost estimate h is used, we would find the neighbor with the lowest h.

Hill-climbing Search Algorithm...

- The hill-climbing search algorithm, which is the most basic local search technique.
- At each step the current node is replaced by the best neighbor.

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
    current ← problem.INITIAL
    while true do
        neighbor ← a highest-valued successor state of current
        if VALUE(neighbor) ≤ VALUE(current) then return current
        current ← neighbor
```

Hill-climbing Search Algorithm...

- The hill-climbing search algorithm, which is the most basic local search technique.
- At each step the current node is replaced by the best neighbor.

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
    current ← problem.INITIAL
    while true do
        neighbor ← a highest-valued successor state of current
        if VALUE(neighbor) ≤ VALUE(current) then return current
        current ← neighbor
```

Simple Hill Climb Algorithm

- Simple hill climbing is the simplest way to implement a hill climbing algorithm.
- It only evaluates the neighbor node state at a time and selects the first one which optimizes current cost and set it as a current state.
- It only checks it's one successor state, and if it finds better than the current state, then move else be in the same state.
- **Features**
- **Less time consuming**
- **Less optimal solution and the solution is not guaranteed**

Simple Hill Climbing

Step 1:Evaluate the Initial State, if Goal- Quit

Step 2:Loop until solution is found or no new operators are left to be applied on the current state.

a)Select and apply an operator(not applied yet) to produce next state)

b)Evaluate new state

i) if new state is Goal->Quit

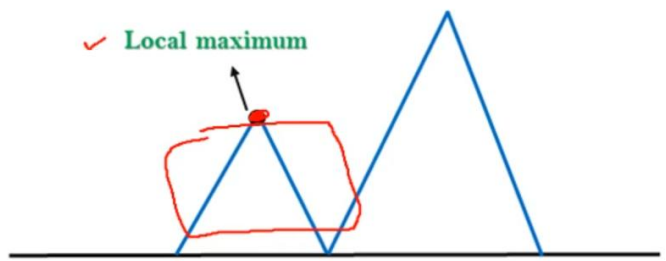
ii)If the new state is better than the current state,assign it as current state

iii)If not better continue in loop

How it moves from current to next state using heuristic function.

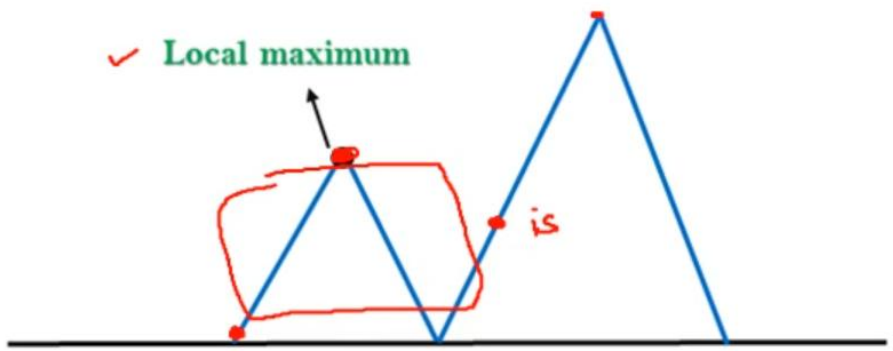
Problems in Hill Climbing Algorithm

- **1. Local Maximum:** A local maximum is a peak state in the landscape which is better than each of its neighboring states, but there is another state also present which is higher than the local maximum.
- **Solution:** Backtracking technique can be a solution of the local maximum in state space landscape.
- Create a list of the promising path so that the algorithm can backtrack the search space and explore other paths as well.



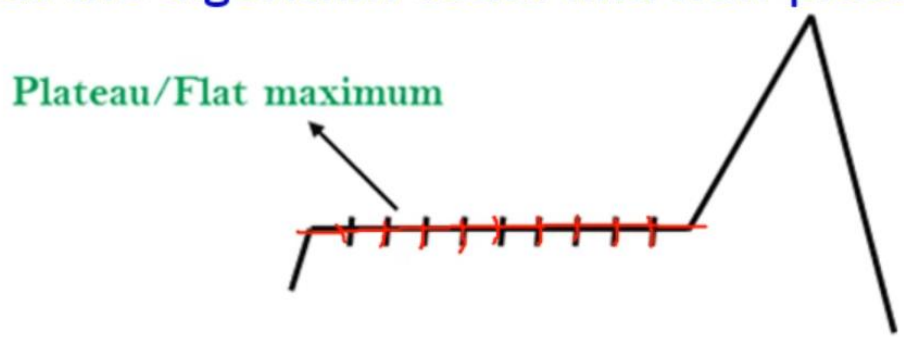
Problems in Hill Climbing Algorithm

- **1. Local Maximum:** A local maximum is a peak state in the landscape which is better than each of its neighboring states, but there is another state also present which is higher than the local maximum.
- **Solution:** Backtracking technique can be a solution of the local maximum in state space landscape.
- Create a list of the promising path so that the algorithm can backtrack the search space and explore other paths as well.



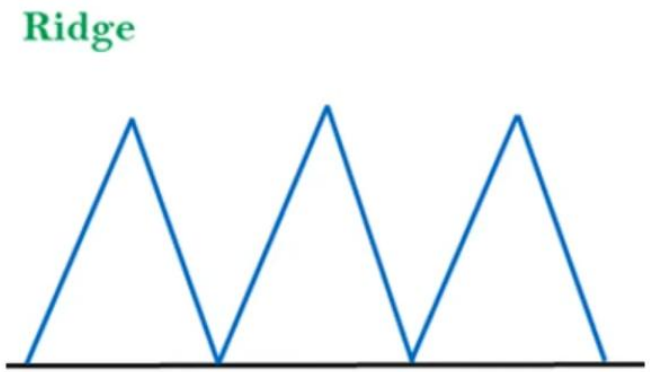
Problems in Hill Climbing Algorithm...

- **2. Plateau:** A plateau is the flat area of the search space in which all the neighbor states of the current state contains the same value, because of this algorithm does not find any best direction to move.
- A hill-climbing search might be lost in the plateau area.
- **Solution:** The solution for the plateau is to take big steps while searching, to solve the problem.
- Randomly select a state which is far away from the current state so it is possible that the algorithm could find non-plateau region.



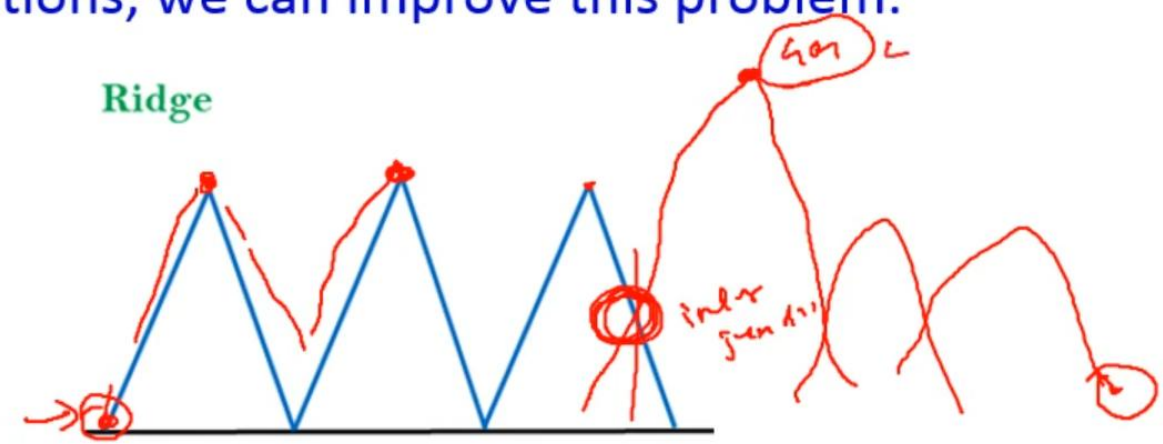
Problems in Hill Climbing Algorithm...

- **3. Ridges:** A ridge is a special form of the local maximum.
- It has an area, which is higher than its surrounding areas, but itself has a slope, and cannot be reached in a single move.
- **Solution:** With the use of bidirectional search, or by moving in different directions, we can improve this problem.



Problems in Hill Climbing Algorithm...

- **3. Ridges:** A ridge is a special form of the local maximum.
- It has an area, which is higher than its surrounding areas, but itself has a slope, and cannot be reached in a single move.
- **Solution:** With the use of bidirectional search, or by moving in different directions, we can improve this problem.



Applications of hill climbing algorithm

- The hill-climbing algorithm can be applied in the following areas:
- **Marketing**
- A hill-climbing algorithm can help a marketing manager to develop the best marketing plans. This algorithm is widely used in solving Traveling-Salesman problems. It can help by optimizing the distance covered and improving the travel time of sales team members. The algorithm helps establish the local minima efficiently.
- **Robotics**
- Hill climbing is useful in the effective operation of robotics. It enhances the coordination of different systems and components in robots.
- **Job Scheduling**
- The hill climbing algorithm has also been applied in job scheduling. This is a process in which system resources are allocated to different tasks within a computer system. Job scheduling is achieved through the migration of jobs from one node to a neighboring node. A hill-climbing technique helps establish the right migration route.

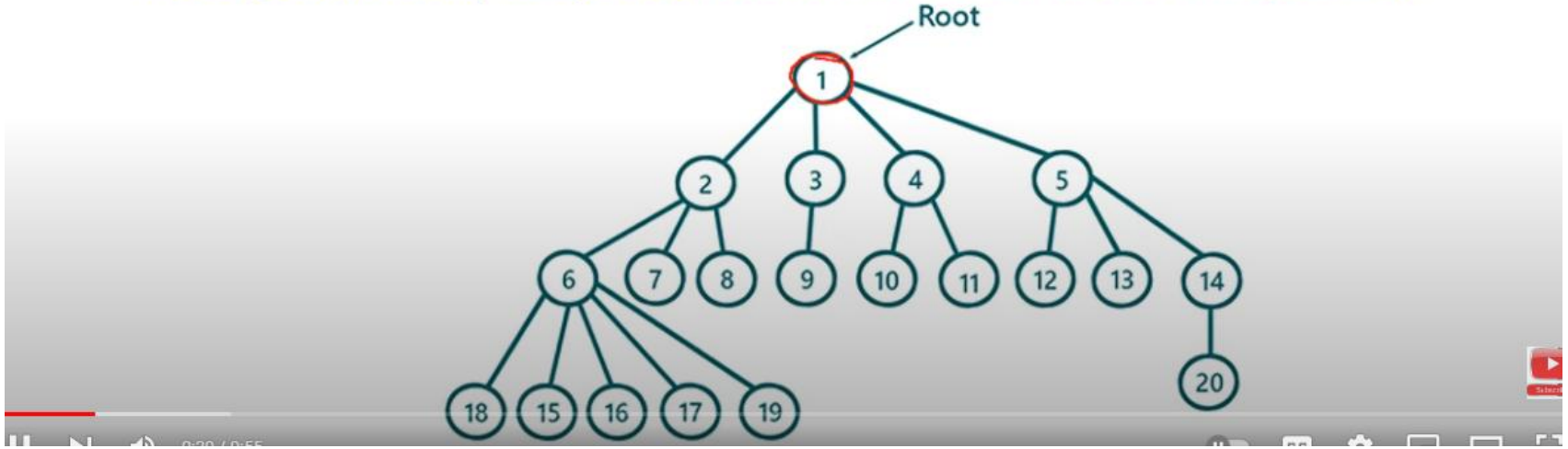
Local Beam Search

- Store only single state in memory
- We cannot guarantee that local sate will provide solution.
- Give a pbm to n members then at least one member will provide solution.

Beam Search Algorithm

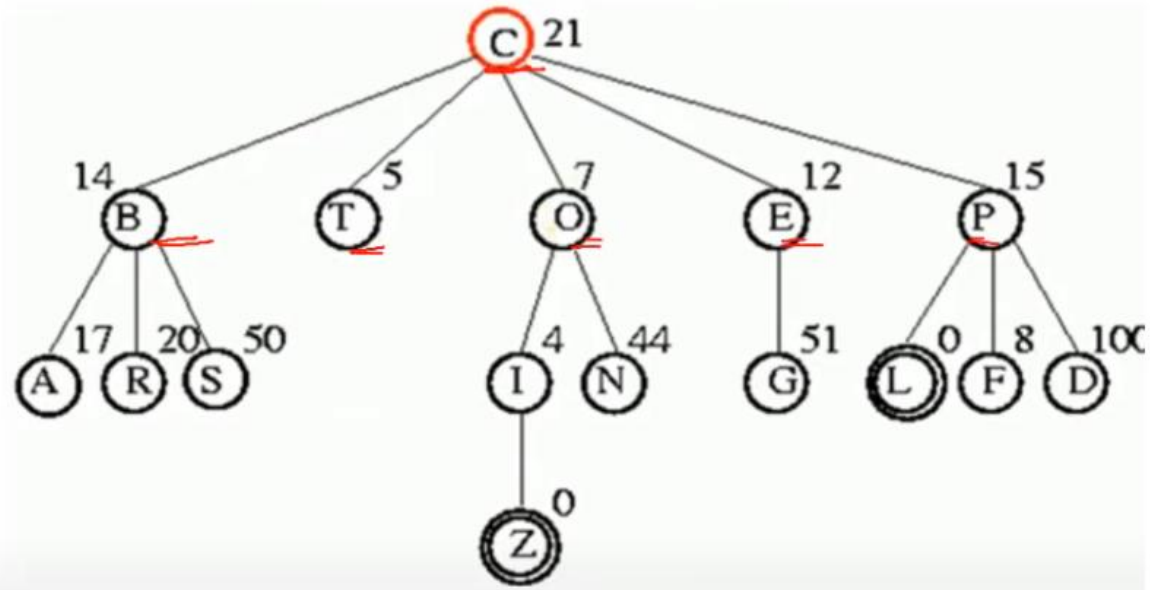
Beam Search Algorithm

- A heuristic search algorithm that examines a graph by extending the most promising node in a limited set is known as **beam search algorithm.**
- The number of nodes n represents the **beam width.**
- This algorithm only keeps the lowest number of nodes on open list,



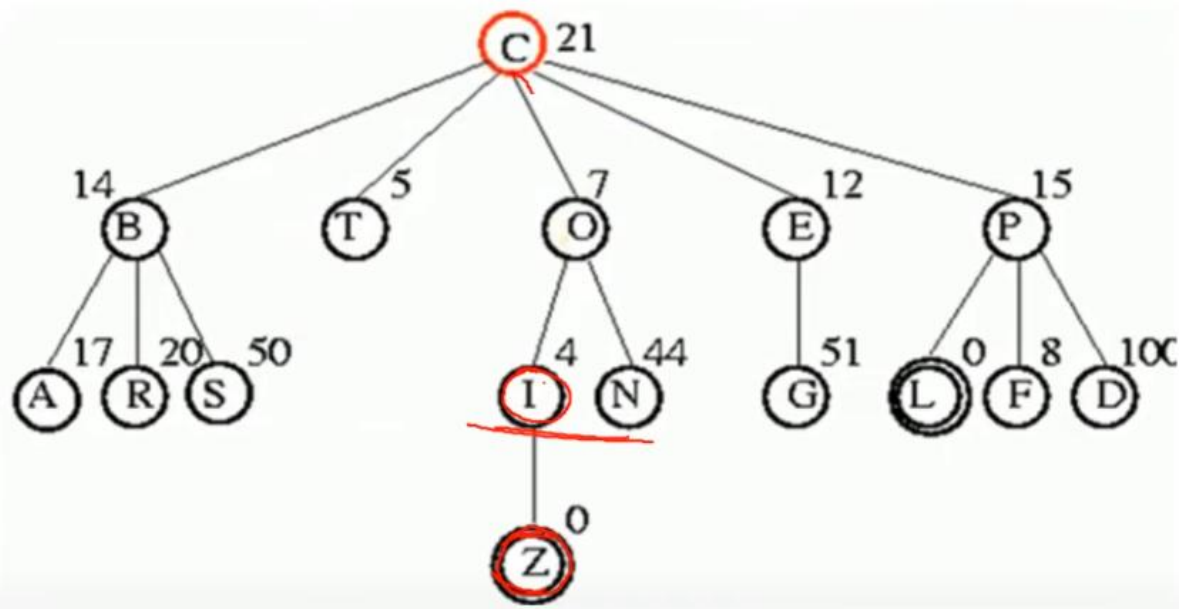
Components of Beam Search

- A beam search takes three components as its input:
- 1. The problem usually represented as graph and contains a set of nodes in which one or more of the nodes represents a goal.
- 2. The set of heuristic rules for pruning: are rules specific to the problem domain and prune unfavorable nodes from memory regarding the problem domain.
- 3. A memory with a limited available capacity
- The memory is where the ***"beam"*** is stored, memory is full, and a node is to be added to the beam, the most costly node will be deleted, such that the memory limit is not exceeded.

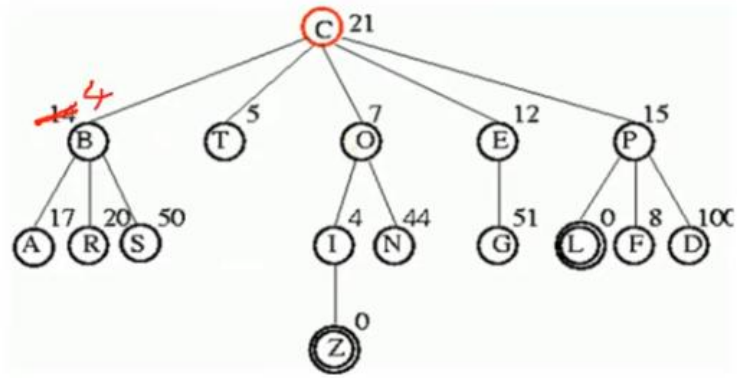


- Iteration 2
- Find successor of C
- = B T O E P
- Remove C from list, now
- Open list = {T,O}
- Iteration 3
- T has no successors, remove T from open list
- Find successor of O, replace O with I and N in open list, then
- Open list = {I, N}

- Open list = {I, N}
- Iteration 4
- Find successor of I
- Z is Goal

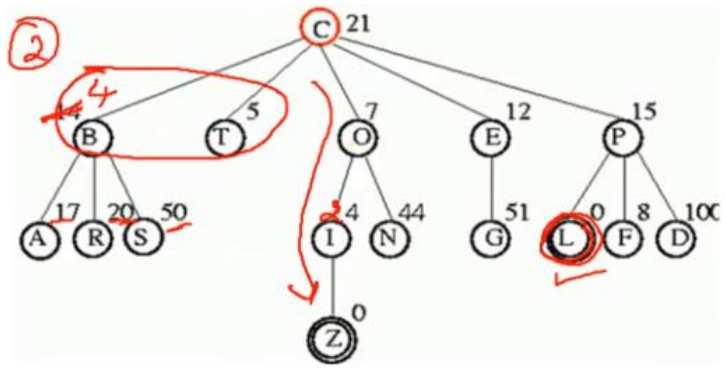


Beam Search Algorithm…



- Beam search algorithm is not complete
  - It is not optimal
  - The time complexity: The worst-case time = ***O(B*m)***
  - The space complexity: The worst-case space complexity = ***O(B*m)***
  - ***B*** is the beam width, and m is the maximum depth of any path in the search tree.

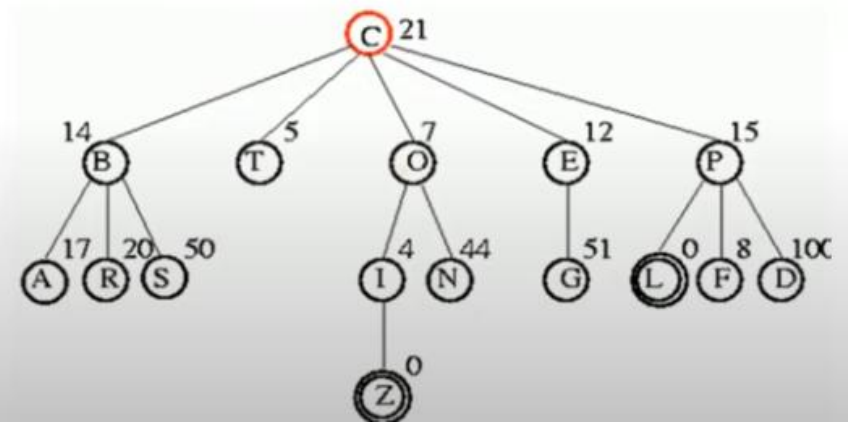
Beam Search Algorithm…



- Beam search algorithm is not complete
- It is not optimal.
  - The time complexity: The worst-case time = ***O(B*m)***
  - The space complexity: The worst-case space complexity = ***O(B*m)***
  - ***B*** is the beam width, and m is the maximum depth of any path in the search tree.

Beam Search Algorithm...

- If the beam size is 1 then this algorithm is Best First Searching
- When compared to A* searching, the time and space complexity will be very much reduced.



Simulated Annealing

Physical Annealing

- The Simulated Annealing algorithm is based upon Physical Annealing in real life.
- Physical Annealing is the process of heating up a material until it reaches an **annealing temperature** and then
- it will be **cooled down** slowly in order to change the material to a desired structure.
- When the material is hot, the molecular structure is weaker and is more susceptible to change.
- When the material cools down, the molecular structure is harder and is less susceptible to change.

Physical Annealing...

- Thermal Dynamics Equation calculates the probability that the Energy Magnitude will increase.

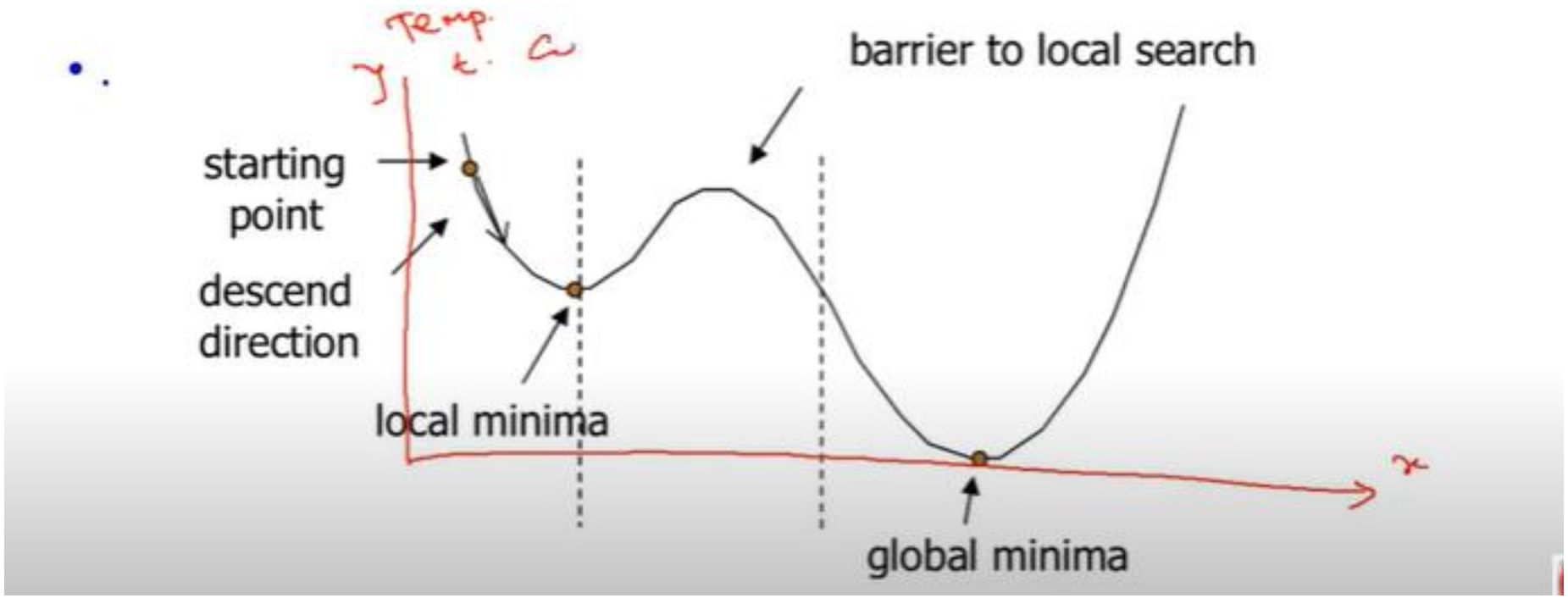
$$P(\Delta E) = e^{-\frac{\Delta E}{k*t}}$$

- Where ΔE - Energy Magnitude
- t - temperature
- k - Boltzmann constant.

Simulated Annealing

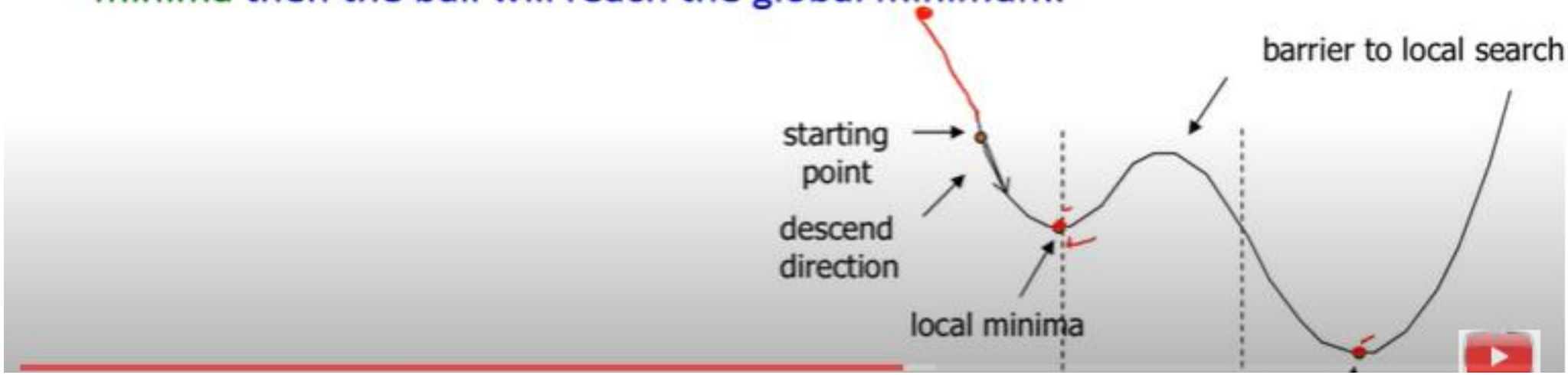
- **Simulated Annealing** is a stochastic global search optimization algorithm and it is modified version of stochastic hill climbing.
- This algorithm appropriate for nonlinear objective functions, where other local search algorithms do not operate well.
- **The simulated-annealing solution is to start by shaking hard (i.e., at a high temperature) and**
- **then gradually reduce the intensity of the shaking (i.e., lower the temperature).**
- Simulated Annealing (SA) is very useful for situations where there are a lot of local minima.

Simulated Annealing - State Space Diagram



Simulated Annealing- Example- ping-pong ball

- Imagine the task of getting a ping-pong ball into the deepest crevice in a very bumpy surface.
- If we just let the ball roll, it will come to rest at a local minimum.
- The trick is to shake just **hard** enough, to bounce the ball out of local minima then the ball will reach the global minimum.



Simulated Annealing...

- Simulated annealing improves this strategy through the introduction of two tricks.
- This algorithm picks a *random* move instead of picking the best move.
- If the move improves the result then it accepts this random move, otherwise it accepts the move with some probability less than 1.

Simulated Annealing...

- Based on annealing, the algorithm uses the equation

$$e^{-\Delta D/T} > R(0, 1),$$

- Where ΔD is Change of distance,
- T is synthetic temperature (control Variable) and
- R(0,1) is a random number in the interval [0,1].

Simulated Annealing Algorithm

function SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state
  current ← *problem*.INITIAL
  for $t = 1$ **to** ∞ **do**
    $T \leftarrow schedule(t)$
    if $T = 0$ **then return** *current*
    next ← a randomly selected successor of *current*
    $\Delta E \leftarrow$ VALUE(*current*) – VALUE(*next*)
    if $\Delta E > 0$ **then** *current* ← *next*
    else *current* ← *next* only with probability $e^{-\Delta E/T}$

Genetic Algorithm

Genetic Algorithm

- A **genetic algorithm** (or **GA**) is a search technique used to find true or approximate solutions.
- Genetic algorithms are categorized as global search heuristics.
- GAs are particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover (also called recombination).

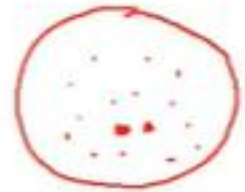
Genetic Algorithm...

- The evolution usually starts from a **population** of randomly generated individuals and happens in generations.
- In each generation, the fitness of every individual in the population is evaluated,
- multiple individuals are selected from the current population (based on their fitness),
- and modified (recombined and possibly mutated) to form a new population.
- The new population is then used in the next iteration of the algorithm.

Steps in Genetic Algorithm

- Select Initial Population
- Fitness function
- Parent selection
- Crossover
- Mutation

Mixing Number



- The mixing number, p, which is the number of parents that come together to form offspring.
- The most common case is p=2: two parents combine their "genes"
- It is possible to have p>2, to simulate on computers.

Selection Process

- The **selection** process for selecting the individuals who will become the parents of the next generation:
- one possibility is to select from all individuals with probability proportional to their fitness score.
- Another possibility is to randomly select n individuals, (n>p) and then select p the most fit ones as parents.

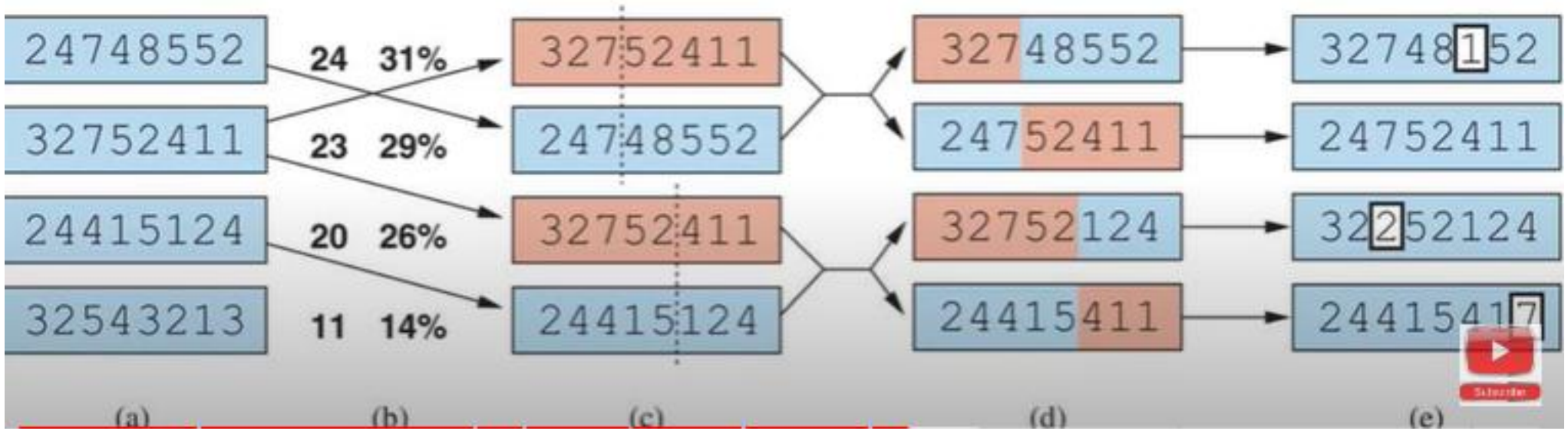
The Recombination Procedure

- It is a random selection of **crossover point** to split each of the parent strings,
- and recombine the parts to form two children,
  - one with the first part of parent 1 and the second part of parent 2;
  - the other with the second part of parent 1 and the first part of parent 2.

Genetic algorithm for 8 queens problem

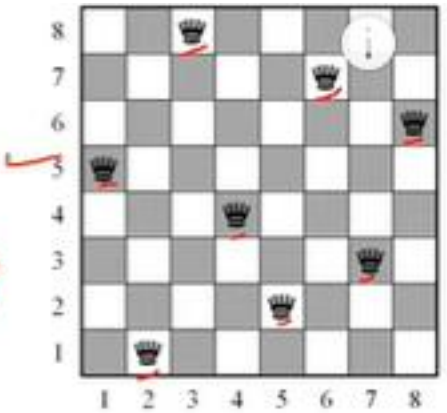
- A genetic algorithm, illustrated for digit strings representing 8-queens states. The initial population in (a) is ranked by a fitness function in (b) resulting in pairs for mating in (c). They produce offspring in (d), which are subject to mutation in (e).

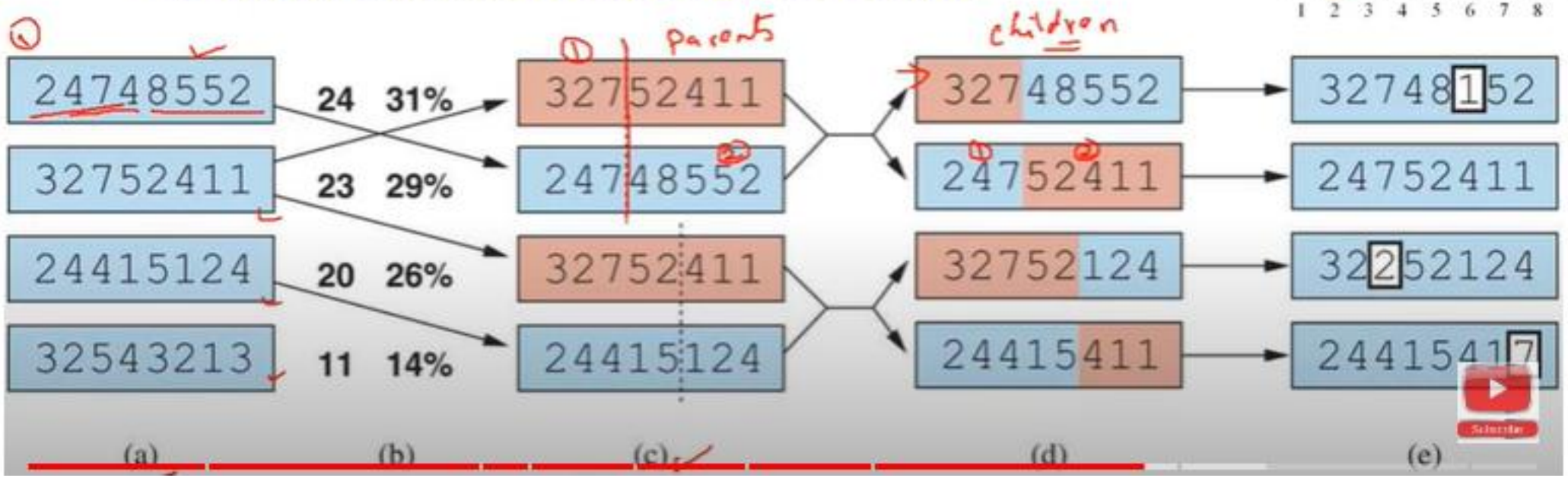
8Q



Genetic algorithm for 8 Queens problem

- A genetic algorithm, illustrated for digit strings representing 8-queens states. The initial population in (a) is ranked by a fitness function in (b) resulting in pairs for mating in (c). They produce offspring in (d), which are subject to mutation in (e).





Constraint Satisfaction Problem

- Constraint satisfaction problems (CSPs) need solutions that satisfy all the associated constraints

CSP

- A constraint satisfaction problem (CSP) is a problem that requires its solution within some limitations or conditions also known as constraints. It consists of the following:

- A finite set of variables which stores the solution (V = {V1, V2, V3,....., Vn})
- A set of discrete values known as domain from which the solution is picked (D = {D1, D2, D3,.....,Dn})
- A finite set of constraints (C = {C1, C2, C3,....., Cn})

Constraint Satisfaction Problem

- Constraint programming or constraint solving is about finding values for variables such that they satisfy a constraint(conditions).
- CSP = {V, D, C}

  Variables: V= {V1,..,Vn}

  Domain: D= { D1,D2,..Dn }

  Constraints: C= {C1,..,Ck}
- **Example:**

  Crossword puzzle

  Crypt-Arithmatic problem

  Map colouring problem

Constraint Satisfaction Problems (CSP)

- A problem that requires its solution within some **limitations/conditions** also known as **constraints**.
- It consists of:

Continuous

- X is a **finite** set of variables, {X1, . . . ,Xn}.
- D is a set of **domains**, {D1, . . . ,Dn}, one for each variable.

Discrete

- C is a **finite** set of constraints that specify allowable combinations of values. (C = {C1, C2, C3,....., Cn})

Constraint Satisfaction Problems

- CSP:
  - state is defined by variables X_i with values from domain D_i
  - goal test is a set of constraints specifying allowable combinations of values for subsets of variables

- Allows useful general-purpose algorithms with more power than standard search algorithms

KHWAJA FAREED UEIT

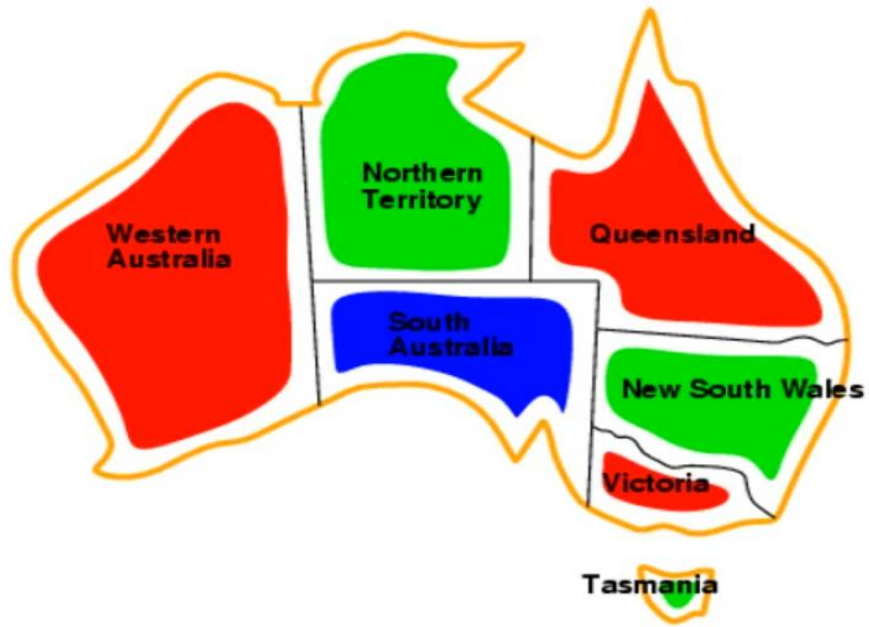
Example of CSP – Map Coloring problem

- The problem is to color the regions of a a given map such that no two adjacent regions have the same colors
- The regions in the map are the variables and the set of possible colors for region is the domain.

- Representation of Problem
- Variables: Vi = {WA, NT, Q, NSW, V, SA, T}
- Domains: Di = {red, green, blue}
- Constraints: adjacent regions must have different colors
- e.g., WA ≠ NT
- —So (WA,NT) must be in {(red,green),(red,blue),(green,red), ...}

- Initial state:
  - The empty assignment {}, in which all variables are unassigned
- Successor Functions
  - A value can be assigned to any unassigned variable, provided that it does not conflict with previously assigned value
- Goal test
  - Current assignment is complete and test whether the goal reached
- Path cost
  - Compute cost for every step

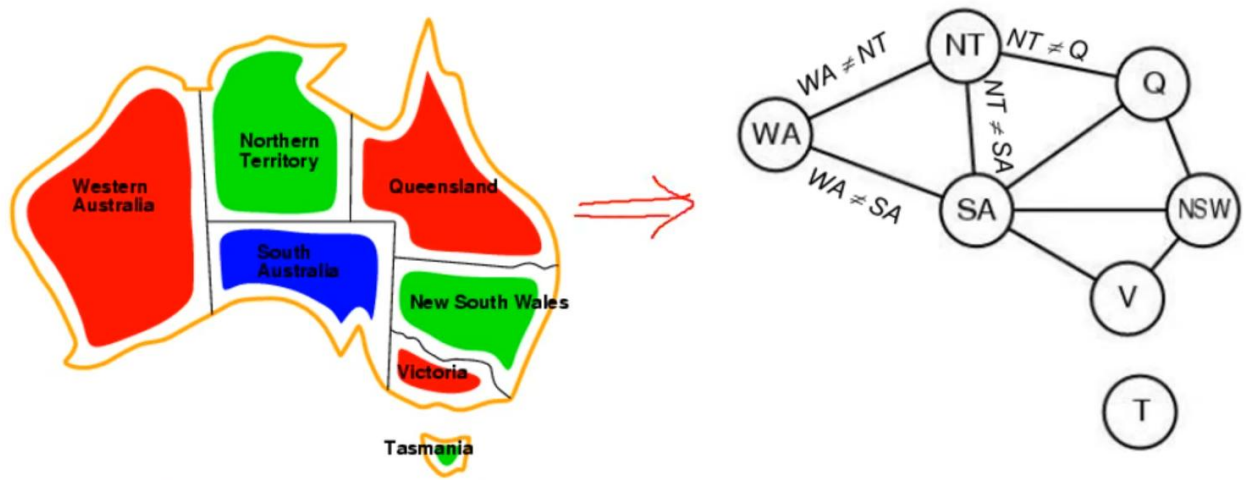
- Solutions are complete and consistent assignments,
- e.g., WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green



CSP: Constraint Graph

- It is helpful to visualize a CSP as a **constraint graph**
- The nodes of the graph correspond to variables of the problem
- Link connects any two variables that participate in a constraint

CSP – Map Coloring Vs Graph Coloring



- Most of the times, these problems has many solution
- The topology of constraint graph can be used to identify solutions easily
- Map coloring can be some times represented as graph coloring problem

Types of constraints in CSP

- Unary constraint (related to single variable in a problem)
- Binary constraint (relates to two variables - Graph Coloring)
- Higher-order constraint (related to many variables - Cryptrithetic puzzle)

Converting Problems to CSP

- **Step 1:** Create a variable set.
- **Step 2:** Create a domain set.
- **Step 3:** Create a constraint set with variables and domains (if possible) after considering the constraints.
- **Step 4:** Find an optimal solution.

Definition: Constraint Satisfaction Problem

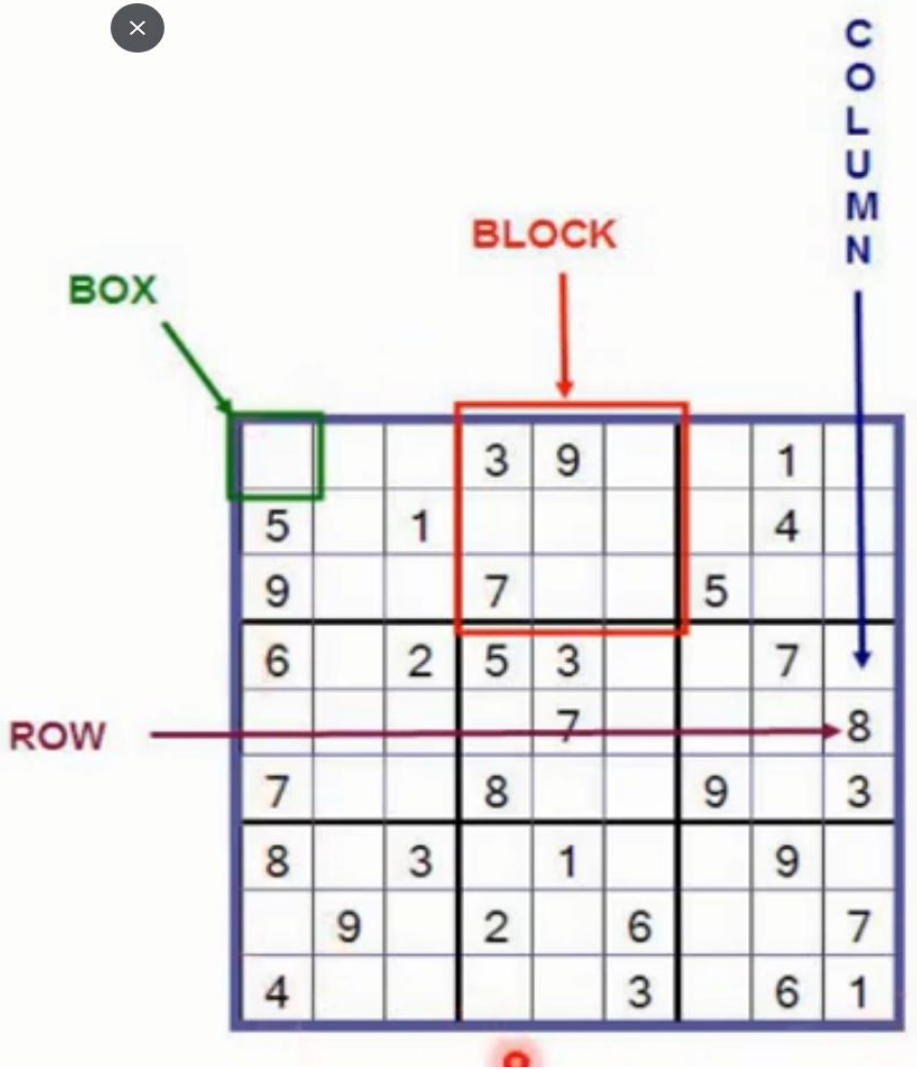
- Consider a Sudoku game with some numbers filled initially in some squares.
- You are expected to fill the empty squares with numbers ranging from 1 to 9 in such a way that no row, column or a block has a number repeating itself.
- This is a very basic constraint satisfaction problem.
- You are supposed to solve a problem keeping in mind some constraints.
- The remaining squares that are to be filled are known as variables, and the range of numbers (1-9) that can fill them is known as a domain. Variables take on values from the domain.
- The conditions governing how a variable will choose its domain are known as constraints.

CSP: Example SUDOKO

Variables ->

Domain ->

Constraints ->



KHWAJA FAREED UEIT RAHIM YAR KHAN

CSP: Example
SUDOKO

Variables ->

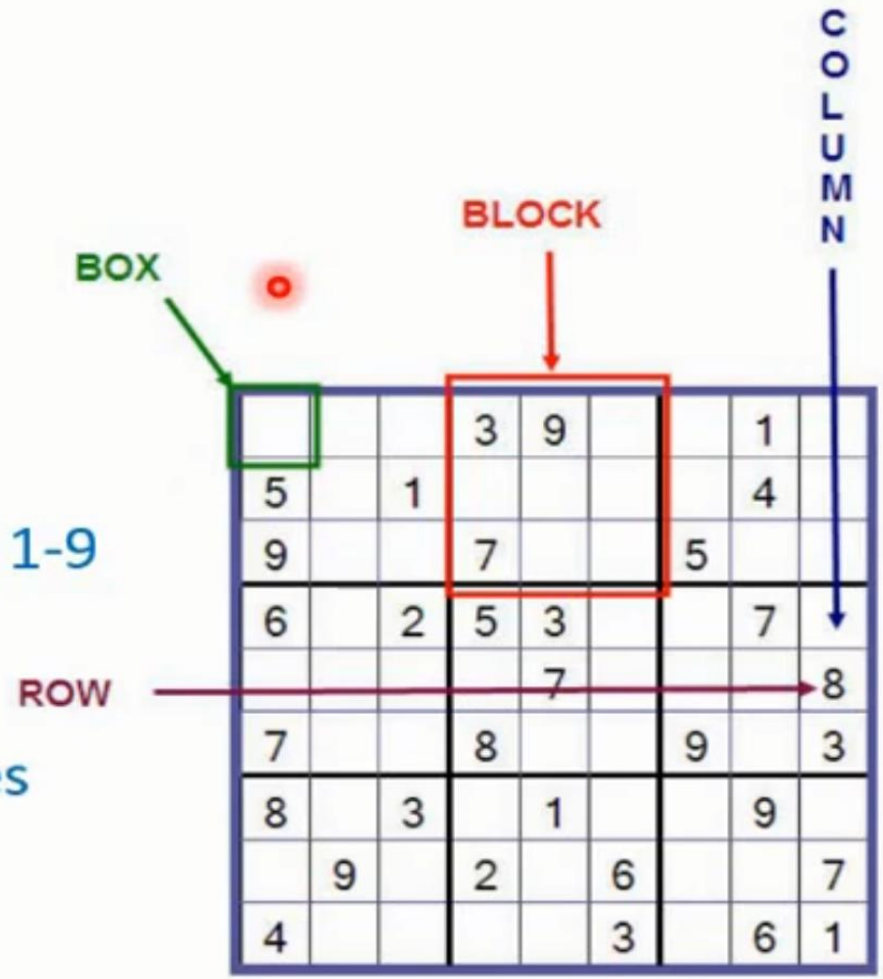
Boxes

Domain ->

Range of Numbers 1-9

Constraints ->

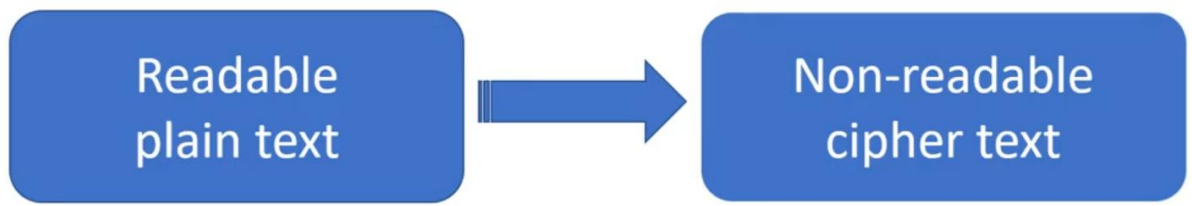
Game Rules

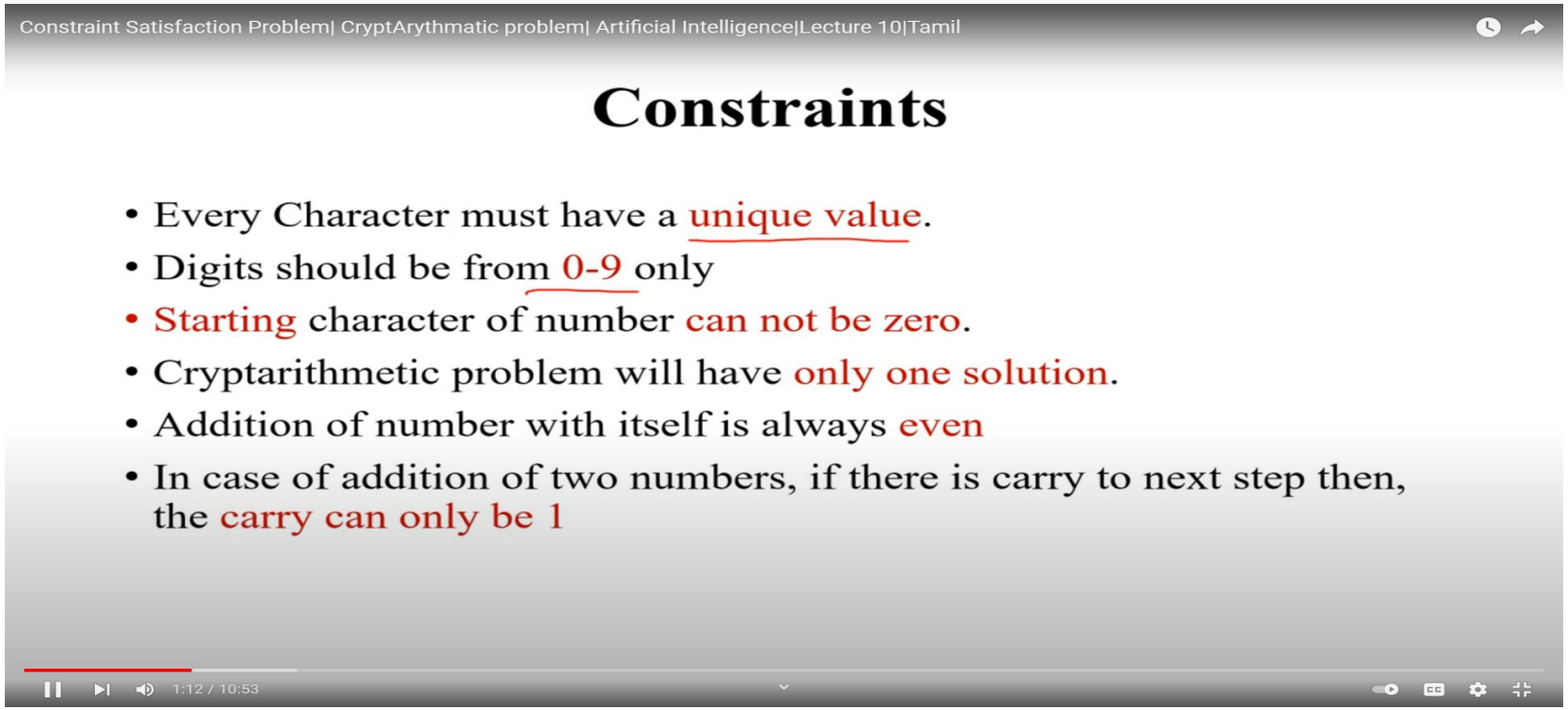


KHWAJA FAREED UEIT RAHIM YAR KHAN

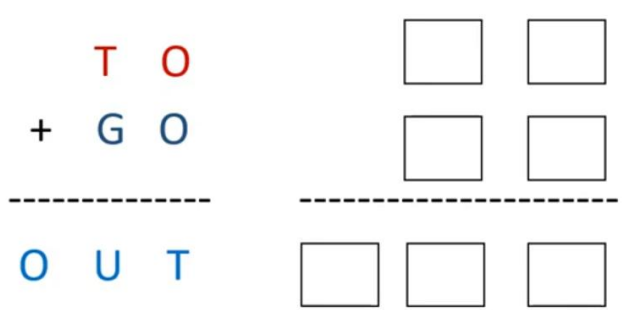
Crypt-Arithmetic problem

- Crypt-Arithmetic problem is a type of **encryption problem** in which the written message in an alphabetical form which is easily readable and understandable is converted into a numeric form which is neither easily readable nor understandable.




Constraint Satisfaction Problem| CryptArythmatic problem| Artificial Intelligence|Lecture 10|Tamil
Constraints
• Every Character must have a unique value.
• Digits should be from 0-9 only
• Starting character of number can not be zero.
• Cryptarithmetic problem will have only one solution.
• Addition of number with itself is always even
• In case of addition of two numbers, if there is carry to next step then, the carry can only be 1
1:12 / 10:53

Example

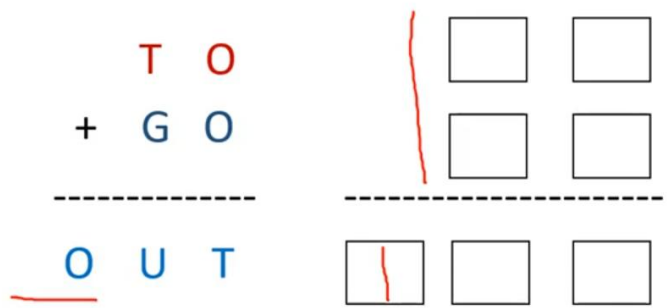


T	
O	
G	
U	

- Variables :T,O,G,U
- Domains :0, 1, 2, 3 ,4, 5, 6, 7, 8, 9

+

Example

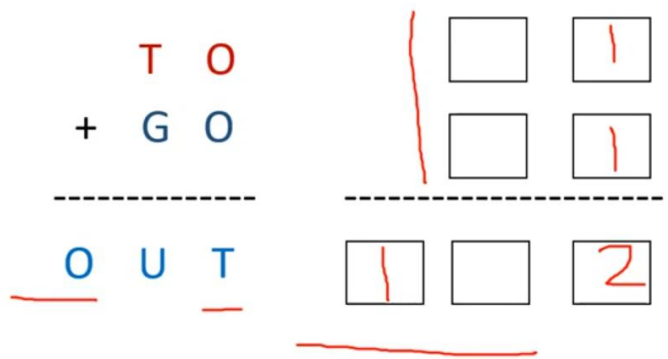


T	
O	
G	
U	

- Variables :T,O,G,U
- Domains :0, 1, 2, 3 ,4, 5, 6, 7, 8, 9

+

Example



T	2
O	1
G	
U	

- Variables :T,O,G,U
- Domains :0, 1, 2, 3 ,4, 5, 6, 7, 8, 9

+

Example

2 + 8 = 10

```
  T  O
+ G  O
-------
O  U  T
```

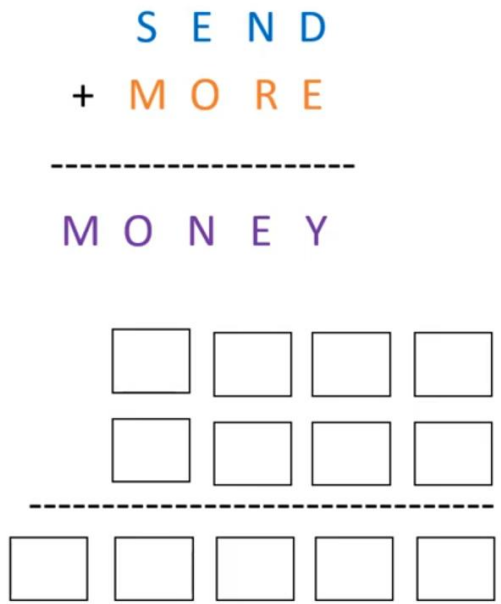
2	1
8	1
1	C	2

T	2
O	1
G	
U	

- Variables :T,O,G,U
- Domains :0, 1, 2, 3 ,4, 5, 6, 7, 8, 9

+

Example



S	
M	
E	
O	
N	
R	
D	
Y	

Popular Problems of CSP

- CryptArithmetic
- n-Queen
- Map Coloring
- Crossword
- Sudoku
- Latin Square Problem