MODULE 3

SEARCH IN COMPLEX ENVIRONMENTS

EDULINE FOR CSE STUDENTS

Prepared By Mr. EBIN PM, Chandigarh University, Punjab 1

ADVERSARIAL SEARCH

- In artificial intelligence, deep learning, machine learning, adversarial search is basically a kind of search in which one can trace the movement of an enemy or opponent.
- Adversarial search is a method applied to a situation where you are planning while another actor prepares against you.
- Your plans, therefore, could be affected by your opponent's actions. The term "search" in the context of adversarial search refers to games, at least in the realm of Artificial Intelligence.
- Adversarial search problems typically exist in two-player games where the players' actions alternate.

Prepared By Mr. EBIN PM, Chandigarh University, Punjab EDULINE 2

- Examples are chess, checkers, and tic-tac-toe.
- In the chess game your next moves, however, would entirely depend on your opponent's moves. The other player's countermoves, on the other hand, would also be dependent on your opening moves, and so on.
- In games, the AI agent has been surrounded by a kind of competitive environment. The goal has been defined initially by the user and the agents compete or fight with one another in order to achieve that goal so that the win can be achieved.
- The adversarial search is important, and each agent must have known the strategy of the next agent. This will create a competitive environment in a game.

- **Searches in which two or more players with conflicting goals are trying to explore the same search space for the solution, are called adversarial searches, often known as Games.**
- Games are modeled as a Search problem and heuristic evaluation function, and these are the two main factors which help to model and solve games in AI.

❖**Types of Games in AI:**

1. **Perfect information:** A game with the perfect information is that in which agents can look into the complete board. Agents have all the information about the game, and they can see each other moves also. Examples are Chess, Checkers, Go, etc.

2. **Imperfect information:** If in a game agents do not have all information about the game and not aware with what's going on, such type of games are called the game with imperfect information, such as tic-tac-toe, Battleship, blind, Bridge, etc.
3. **Deterministic games:** Deterministic games are those games which follow a strict pattern and set of rules for the games, and there is no randomness associated with them. Examples are chess, Checkers, Go, tic-tac-toe, etc.
4. **Non-deterministic games:** Non-deterministic are those games which have various unpredictable events and has a factor of chance or luck. This factor of chance or luck is introduced by either dice or cards. These are random, and each action response is not fixed. Such games are also called as stochastic games. Example: Backgammon, Monopoly, Poker, etc.

➢ **A game can be formalized of the following elements:**

- **Initial state:** It specifies how the game is set up at the start.
- **Player(s):** It specifies which player has moved in the state space.
- **Action(s):** It returns the set of legal moves in state space.
- **Result(s, a):** It is the transition model, which specifies the result of moves in the state space.
- **Terminal-Test(s):** Terminal test is true if the game is over, else it is false at any case. The state where the game ends is called terminal states.
- **Utility(s, p):** A utility function gives the final numeric value for a game that ends in terminal states s for player p. It is also called payoff function. For Chess, the outcomes are a win, loss, or draw and its payoff values are +1, 0, ½. And for tic-tac-toe, utility values are +1, -1, and 0.

❖GAME TREE

- A game tree is a tree where nodes of the tree are the game states and Edges of the tree are the moves by players. Game tree involves initial state, actions function, and result Function.

❖TECHNIQUES REQUIRED TO GET THE BEST OPTIMAL SOLUTION

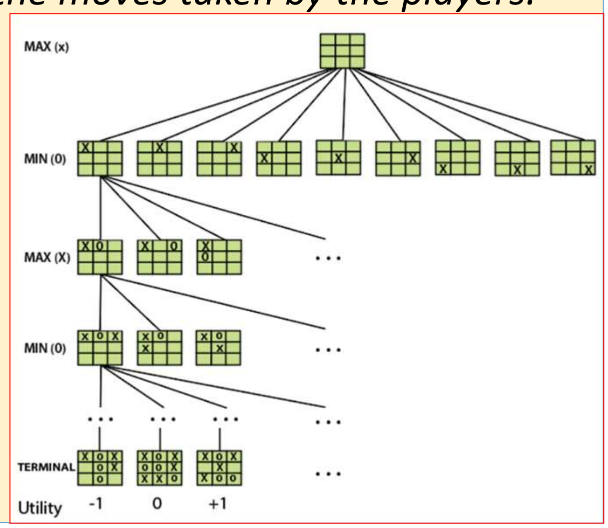
- There is always a need to choose those algorithms which provide the best optimal solution in a limited time. So, we use the following techniques which could fulfill our requirements:

➢**Pruning:** A technique which allows ignoring the unwanted portions of a search tree which make no difference in its final result.

➢**Heuristic Evaluation Function:** It allows to approximate the cost value at each level of the search tree, before reaching the goal node.

- Let's understand the working of the elements with the help of a game tree designed for **tic-tac-toe**. Here, the *node represents the game state and edges represent the moves taken by the players.*

- **A game-tree for tic-tac-toe**



- **INITIAL STATE (S_0):** The top node in the game-tree represents the initial state in the tree and shows all the possible choice to pick out one.
- **PLAYER (s):** There are two players, **MAX** and **MIN**. **MAX** begins the game by picking one best move and place **X** in the empty square box.
- **ACTIONS (s):** Both the players can make moves in the empty boxes chance by chance.
- **RESULT (s, a):** The moves made by **MIN** and **MAX** will decide the outcome of the game.
- **TERMINAL-TEST(s):** When all the empty boxes will be filled, it will be the terminating state of the game.
- **UTILITY:** At the end, we will get to know who wins: **MAX** or **MIN,** and accordingly, the price will be given to them**. (-1):** If the PLAYER loses. **(+1):** If the PLAYER wins. **(0):** If there is a draw between the PLAYERS.

❖Types of algorithms in Adversarial search

- In a normal search, we follow a sequence of actions to reach the goal or to finish the game optimally. But in an **adversarial search**, the result depends on the players which will decide the result of the game.
- It is also obvious that the solution for the goal state will be an optimal solution because the player will try to win the game with the shortest path and under limited time.
- There are following types of adversarial search:

1. **Minmax Algorithm**
2. **Alpha-beta Pruning**

MIN MAX ALGORITHM

- Min-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally.
- Min-Max algorithm uses recursion to search through the game-tree.
- Min-Max algorithm is mostly used for game playing in AI, such as Chess, Checkers, tic-tac-toe, go etc.
- This Algorithm computes the min-max decision for the current state.

- In this algorithm two players play the game, one is called **MAX** and other is called **MIN**.
- MAX will select the maximized value
- MIN will select the minimized value.
- The min-max algorithm performs a depth-first search algorithm for the exploration of the complete game tree.
- The min-max algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion

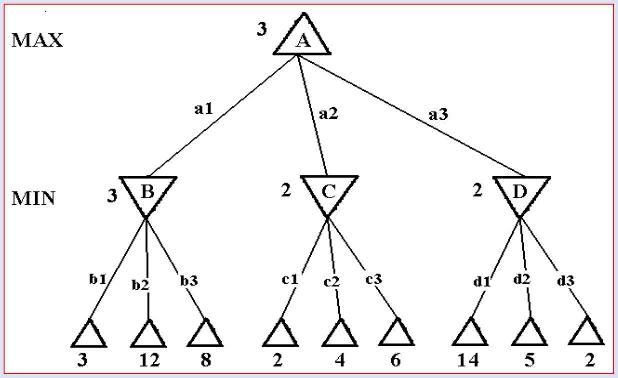
❖Minimax search procedure

- Consider games with 2 players. The opponents in a game are referred to as MIN and MAX.
- MAX represents the player trying to win or to MAXimize his advantage.
- MIN is the opponent who attempts to MINimize MAX's score.
- MAX moves first, and then they take turns moving until the game is over.
- At the end of the game, points are awarded to the winning player and penalties are given to the loser.

❖Search strategies

Consider the game tree

- The ▲ nodes are MAX nodes, in which it is MAX's turn to move and the nodes ▼ are MIN nodes. The terminal states show the utility values for MAX.
- The possible moves for MAX at the root node are labeled a1, a2, a3. the possible replies to a1 for MIN are b1, b2, b3 and so on. This game ends after one move each by MAX and MIN.
- Given a game tree, the optimal strategy can be determined by examining the min-max value of each node, which we write as minmax-value (n).
- The minmax-value of a node is the utility for MAX of being in the corresponding state.
- The minmax-value of a terminal state is just its utility.



MAX will prefer to move to a state of maximum value, whereas MIN prefers a state of minimum value.

So we have

$$\text{Minimax-value}(n) = \begin{cases} \text{Utility}(n), & \textbf{if n is a terminal state} \\ \text{MAX}_{s \in successors(n)}\ \text{minimax-value}(s), & \textbf{If n is a MAX node} \\ \text{MIN}_{s \in successors(n)}\ \text{minimax-value}(s), & \textbf{If n is a MIN node} \end{cases}$$

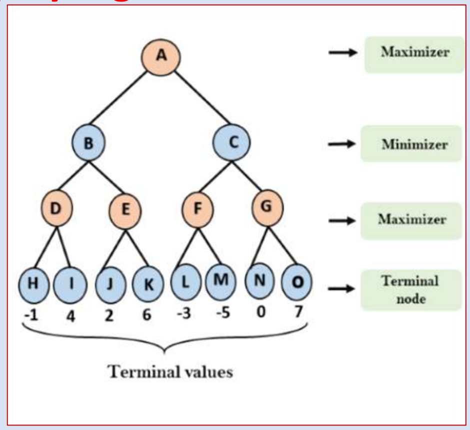
- **Minmax-value(A) = max [min (3,12,8), min (2,4,6), min (14,5,2)]**
  = max [3, 2, 2]
  = 3

❖Working of Min-Max Algorithm

- The working of the minmax algorithm can be easily described using an example. Below we have taken an example of game-tree which is representing the two-player game.
- In this example, there are two players one is called Maximizer and other is called Minimizer.
- Maximizer will try to get the Maximum possible score, and Minimizer will try to get the minimum possible score.
- This algorithm applies DFS, so in this game-tree, we have to go all the way through the leaves to reach the terminal nodes.
- At the terminal node, the terminal values are given so we will compare those value and backtrack the tree until the initial state occurs.

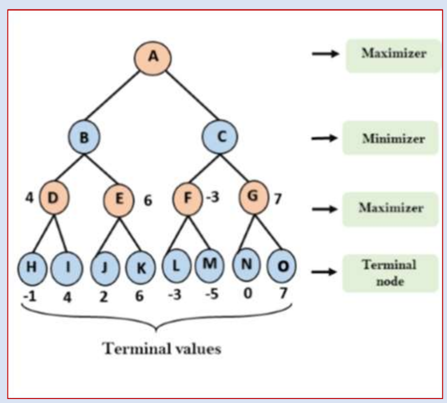
Steps involved in solving the two-player game tree

- **Step-1:** In the first step, the algorithm generates the entire game-tree and apply the utility function to get the utility values for the terminal states.
- In the tree diagram, let's take A is the initial state of the tree.

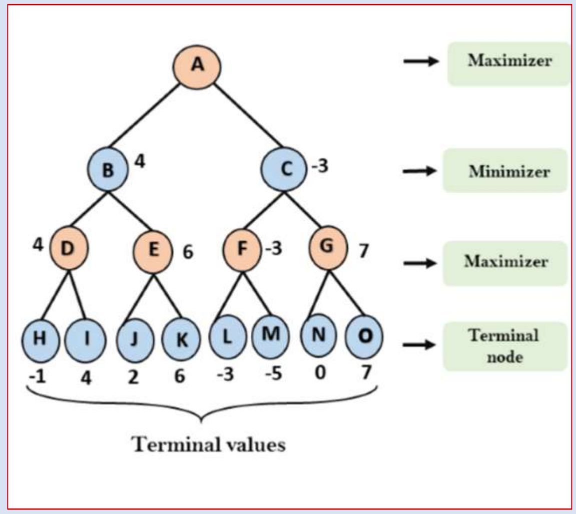


- Suppose maximizer takes first turn which has worst-case initial value = **-infinity**, and minimizer will take next turn which has worst-case initial value = **+infinity**.

- **Step 2:** Now, first we find the utilities value for the Maximizer, its initial value is -∞, so we will compare each value in terminal state with initial value of Maximizer and determines the higher nodes values. It will find the maximum among the all.
- For node D: max(-1, -∞) => max(-1,4)= 4
- For Node E : max(2, -∞) => max(2, 6)= 6
- For Node F :max(-3, -∞) => max(-3,-5)= -3
- For node G: max(0, -∞) = max(0, 7) = 7

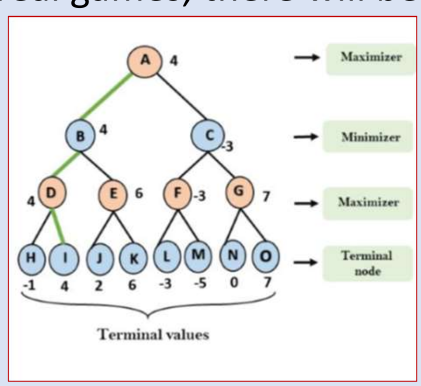


- **Step 3:** In the next step, it's a turn for minimizer, so it will compare all nodes value with +∞, and will find the 3rd layer node values.
- For node B= min(4,6) = 4
- For node C= min (-3, 7) = -3



Prepared By Mr. EBIN PM, Chandigarh University, Punjab EDULINE 19

- **Step 4:** Now it's a turn for Maximizer, and it will again choose the maximum of all nodes value and find the maximum value for the root node.
- In this game tree, there are only 4 layers, hence we reach immediately to the root node, but in real games, there will be more than 4 layers.
- For node A max(4, -3)= 4



Prepared By Mr. EBIN PM, Chandigarh University, Punjab EDULINE 20

❖Properties of Mini-Max algorithm

- **Complete-** Min-Max algorithm is Complete. It will definitely find a solution (if exist), in the finite search tree.
- **Optimal-** Min-Max algorithm is optimal if both opponents are playing optimally.
- **Time complexity-** As it performs DFS for the game-tree, so the time complexity of Min-Max algorithm is **$O(b^m)$**, where b is branching factor of the game-tree, and m is the maximum depth of the tree.
- **Space Complexity-** Space complexity of Mini-max algorithm is also similar to DFS which is **O(bm)**.

❖Limitation of the minimax Algorithm

- The main drawback of the minimax algorithm is that it gets really slow for complex games such as Chess, go, etc.
- This type of games has a huge branching factor, and the player has lots of choices to decide.

ALPHA BETA PRUNING

- Alpha-beta pruning is a modified version of the minmax algorithm.
- It is an optimization technique for the minimax algorithm.
- There is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called **pruning**.
- This involves two threshold parameter Alpha and Beta for future expansion, so it is called **alpha-beta pruning**. It is also called as **Alpha-Beta Algorithm**.
- Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prune the tree leaves but also entire sub-tree.

❖ The two-parameter

- **Alpha:** The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is **-∞**.
- **Beta:** The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is **+∞**.
- The Alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does, but it removes all the nodes which are not really affecting the final decision but making algorithm slow. Hence by pruning these nodes, it makes the algorithm fast.

❖Condition for Alpha-beta pruning:

- The main condition which required for alpha-beta pruning is:

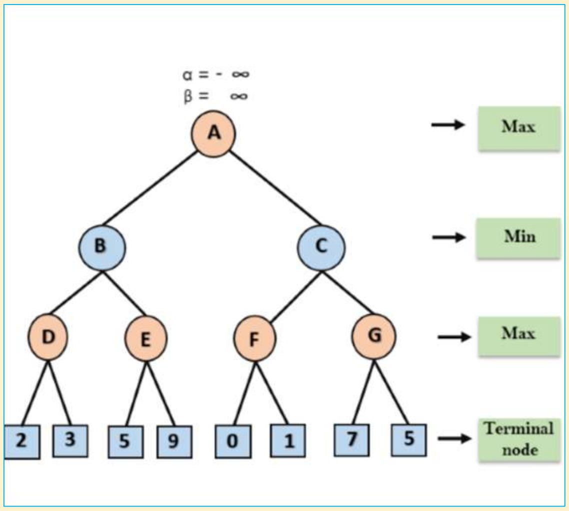
$$\alpha>=\beta$$

❖Key points about alpha-beta pruning

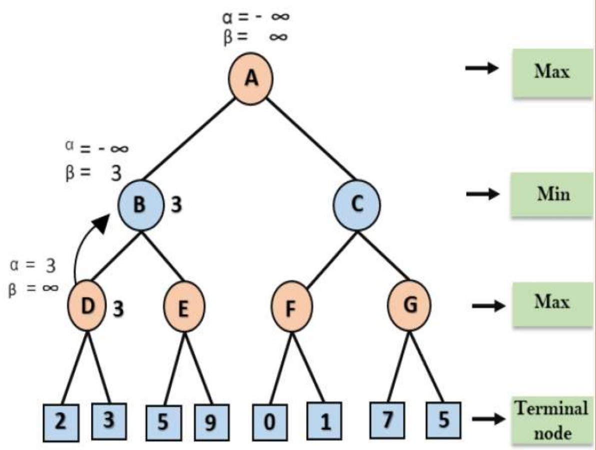
- The Max player will only update the value of alpha.
- The Min player will only update the value of beta.
- While backtracking the tree, the node values will be passed to upper nodes instead of values of alpha and beta.
- We will only pass the alpha, beta values to the child nodes.

❖Working of Alpha-Beta Pruning

- Let's take an example of two-player search tree to understand the working of Alpha-beta pruning
- **Step 1:** At the first step the, Max player will start first move from node A where α= -∞ and β= +∞, these value of alpha and beta passed down to node B where again α= -∞ and β= +∞, and Node B passes the same value to its child D.



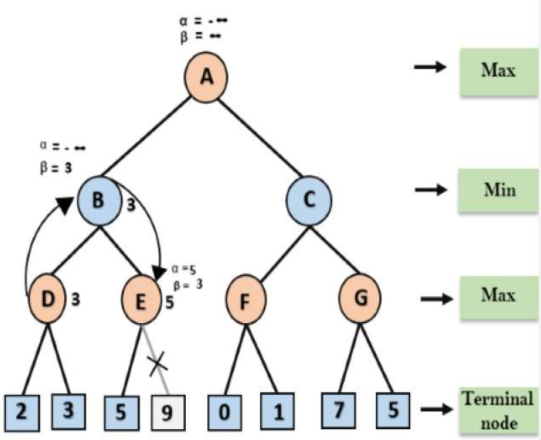
- **Step 2:** At Node D, the value of α will be calculated as its turn for Max. The value of α is compared with firstly 2 and then 3, and the max (2, 3) = 3 will be the value of α at node D and node value will also 3.
- **Step 3:** Now algorithm backtrack to node B, where the value of β will change as this is a turn of Min, Now β= +∞, will compare with the available subsequent nodes value, i.e. min (∞, 3) = 3, hence at node B now α= -∞, and β= 3.



- In the next step, algorithm traverse the next successor of Node B which is node E, and the values of α= -∞, and β= 3 will also be passed.

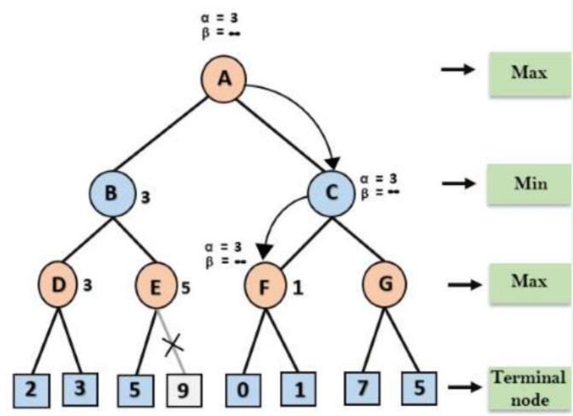
Prepared By Mr. EBIN PM, Chandigarh University, Punjab EDULINE 27

- **Step 4:** At node E, Max will take its turn, and the value of alpha will change. The current value of alpha will be compared with 5, so max (-∞, 5) = 5, hence at node E α= 5 and β= 3, where α>=β, so the right successor of E will be pruned, and algorithm will not traverse it, and the value at node E will be 5.



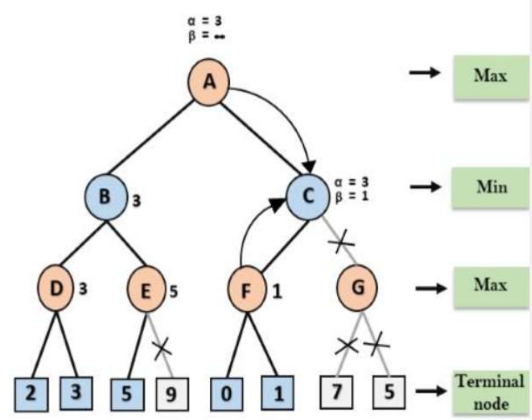
Prepared By Mr. EBIN PM, Chandigarh University, Punjab EDULINE 28

- **Step 5:** At next step, algorithm again backtrack the tree, from node B to node A. At node A, the value of alpha will be changed the maximum available value is 3 as max (-∞, 3)= 3, and β= +∞, these two values now passes to right successor of A which is Node C.
- At node C, α=3 and β= +∞, and the same values will be passed on to node F.
- **Step 6:** At node F, again the value of α will be compared with left child which is 0, and max(3,0)= 3, and then compared with right child which is 1, and max(3,1)= 3 still α remains 3, but the node value of F will become 1.



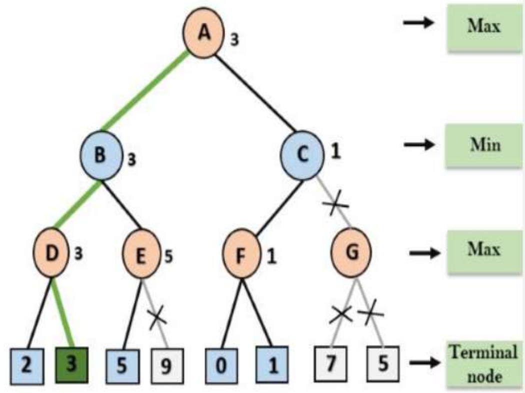
Prepared By Mr. EBIN PM, Chandigarh University, Punjab EDULINE 29

- **Step 7:** Node F returns the node value 1 to node C, at C α= 3 and β= +∞, here the value of beta will be changed, it will compare with 1 so min (∞, 1) = 1. Now at C, α=3 and β= 1, and again it satisfies the condition α>=β, so the next child of C which is G will be pruned, and the algorithm will not compute the entire sub-tree G.



Prepared By Mr. EBIN PM, Chandigarh University, Punjab EDULINE 30

- **Step 8:** C now returns the value of 1 to A here the best value for A is max (3, 1) = 3. Following is the final game tree which is the showing the nodes which are computed and nodes which has never computed. Hence the optimal value for the maximizer is 3 for this example.



Prepared By Mr. EBIN PM, Chandigarh University, Punjab EDULINE 31

❖Ordering in Alpha-Beta Pruning

- The effectiveness of alpha-beta pruning is highly dependent on the order in which each node is examined. Move order is an important aspect of alpha-beta pruning.

It can be of two types:

- **Worst ordering:** In some cases, alpha-beta pruning algorithm does not prune any of the leaves of the tree, and works exactly as minimax algorithm. In this case, it also consumes more time because of alpha-beta factors, such a move of pruning is called worst ordering. In this case, the best move occurs on the right side of the tree. The time complexity for such an order is O($\mathbf{b^m}$).

Prepared By Mr. EBIN PM, Chandigarh University, Punjab EDULINE 32

- **Ideal ordering:** The ideal ordering for alpha-beta pruning occurs when lots of pruning happens in the tree, and best moves occur at the left side of the tree. We apply DFS hence it first search left of the tree and go deep twice as minimax algorithm in the same amount of time. Complexity in ideal ordering is O($b^{m/2}$).

CONSTRAINT SATISFACTION PROBLEMS(CSP)

- 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.

➢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})

➢Please note, that the elements in the domain can be both continuous and discrete but in AI, we generally only deal with discrete values.

❖**Popular Problems with CSP**

- CryptArithmetic (Coding alphabets to numbers.)
- n-Queen (In an n-queen problem, n queens should be placed in an n×n matrix such that no queen shares the same row, column or diagonal.)
- Map Coloring (coloring different regions of map, ensuring no adjacent regions have the same color)
- Crossword (everyday puzzles appearing in newspapers)
- Sudoku (a number grid)

- A constraint is a restriction.

➢ There are many real-life problems that require to give a decision in the presence of constraints:

✓ flight / train scheduling;

✓ scheduling of events in an operating system;

✓ staff rostering at a company;

✓ course time tabling at a university …

- Such problems are called Constraint Satisfaction Problems (CSPs).
- A solution to a CSP is an assignment of values to the variables which satisfies all the constraints simultaneously

❖ **CSP EXAMPLE**

Variables

X = {X1, X2, X3}

Domains

D(X1) = {1,2}, D(X2) = {0,1,2,3}, D(X3) = {2,3}

Constraints

X1 > X2 and X1 + X2 = X3 and X1 ≠ X2 ≠ X3 ≠ X1

Solution

X1 = 2, X2 = 1, X3 = 3

❖CSP EXAMPLE - Job-Shop Scheduling

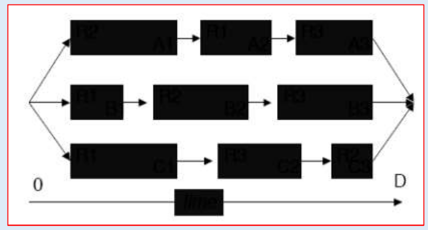
- Schedule jobs, each using a resource for a period, in time D by obeying the precedence and capacity constraints
- A very common industrial problem.

➢ **CSP:**

– variables represent the operations;

– domains represent the start times;

– constraints specify precedence and exclusivity.



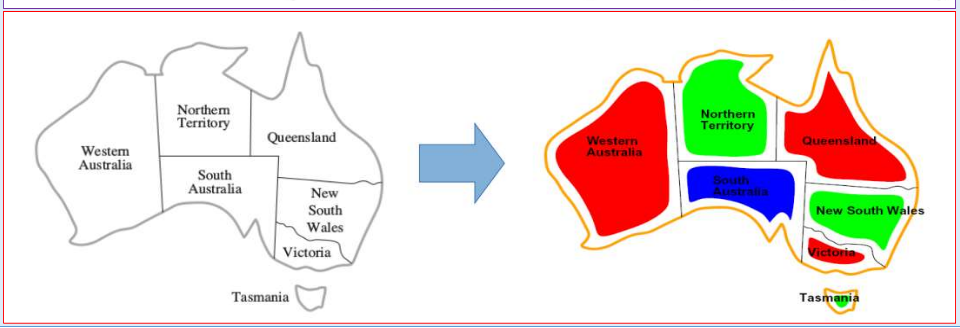
❖CSP EXAMPLE – MAP COLOURING

- **Problem:** assign each territory a color such that no two adjacent territories have the same color

Variables: $X = \{WA, NT, Q, NSW, V, SA, T\}$

Domain of variables: $D = \{r, g, b\}$

Constraints: $C = \{SA \neq WA, SA \neq NT, SA \neq Q, \ldots\}$



Constraint graph: The nodes of the graph correspond to variables of the problem, and a link connects to any two variables that participate in a constraint.

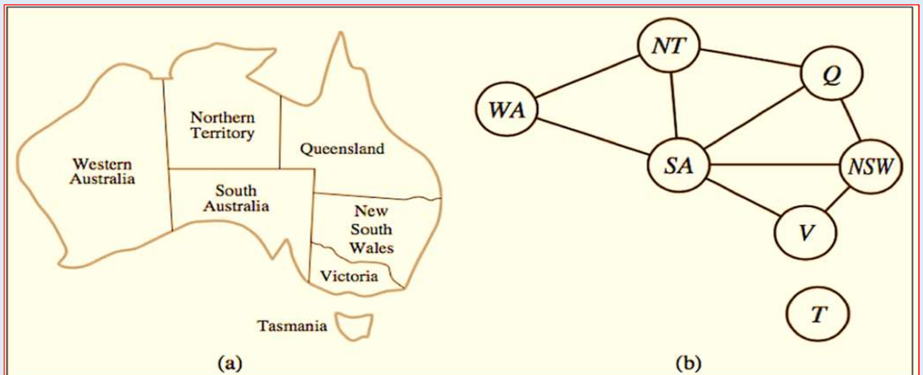


Figure 6.1 (a) The principal states and territories of Australia. Coloring this map can be viewed as a constraint satisfaction problem (CSP). The goal is to assign colors to each region so that no neighboring regions have the same color. (b) The map-coloring problem represented as a constraint graph.

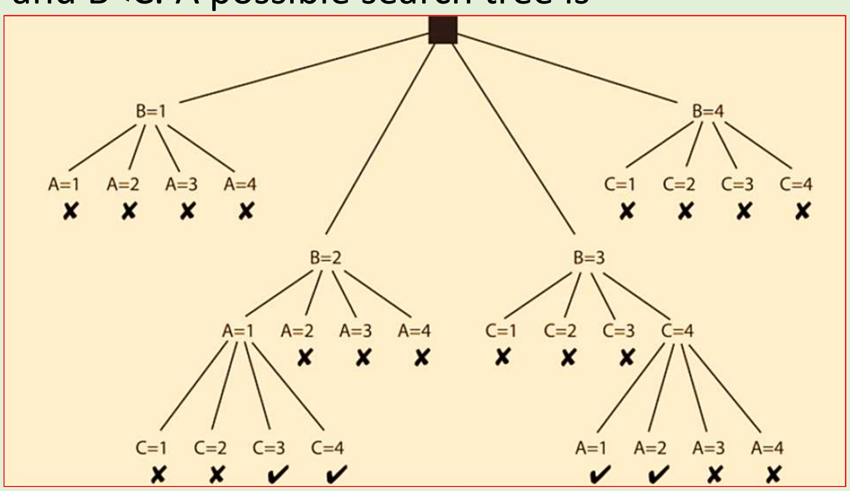
❖Constraint Types in CSP

- With respect to the variables, basically there are following types of constraints:
- **Unary Constraints:** It is the simplest type of constraints that restricts the value of a single variable. A **unary constraint** is a constraint on a single variable [e.g : B≤3 , C(X):X=2 , C(Y): Y>5]
- **Binary Constraints:** It is the constraint type which relates two variables. A value $\mathbf{x_2}$ will contain a value which lies between **x1** and **x3**. A **binary constraint** is a constraint over a pair of variables [e.g : A≤B , C(X,Y): X+Y<6]. It Can be represented by Constraint Graph
- In general, a k-ary constraint has a scope of size k. For example, A+B=C is a 3-ary (ternary) constraint.

➢Some special types of solution algorithms are used to solve the following types of constraints:

- **Linear Constraints:** These type of constraints are commonly used in linear programming where each variable containing an integer value exists in linear form only.
- **Non-linear Constraints:** These type of constraints are used in non-linear programming where each variable (an integer value) exists in a non-linear form.
- A special constraint which works in real-world is known as **Preference constraint.**

- Suppose you have a very simple CSP with the variables A, B, and C, each with domain {1,2,3,4}. Suppose the constraints are A<B and B<C. A possible search tree is



- In this figure, a node corresponds to all of the assignments from the root to that node. The potential nodes that are pruned because they violate constraints are labeled with ✗.
- The leftmost ✗ corresponds to the assignment A=1, B=1. This violates the A<B constraint, and so it is pruned.
- This CSP has four solutions. The leftmost one is A=1, B=2, C=3.
- Searching this tree with a depth-first search, typically called **backtracking.**
- Backtracking is an algorithm that finds all the possible solutions and selects the desired solution from the given set of solutions.

MODEL OF A CSP

- A model of a CSP is an assignment of values to all of its variables that satisfies all of its constraints.

V = {V1}

Dom(V1) = {1,2,3,4}

C = {C1,C2}

- C1: V1 ≠ 2
- C2: V1 > 1

All models for this CSP: **{V1 = 3}**

{V1 = 4}

- A model is a possible world that satisfies all constraints

Another example:

- $\mathcal{V} = \{V_1, V_2\}$
  - $dom(V_1) = \{1,2,3\}$
  - $dom(V_2) = \{1,2\}$
- $\mathcal{C} = \{C_1, C_2, C_3\}$
  - C_1: $V_2 \neq 2$
  - C_2: $V_1 + V_2 < 5$
  - C_3: $V_1 > V_2$

Which are models for this CSP?

$\{V_1=1, V_2=1\}$

$\{V_1=2, V_2=1\}$

$\{V_1=3, V_2=1\}$

$\{V_1=3, V_2=2\}$

❖Scope of a constraint

- The scope of a constraint is the set of variables that are involved in the constraint

Examples:

- V2 ≠ 2 has scope {V2}
- V1 > V2 has scope {V1,V2}
- V1 + V2 + V4 < 5 has scope {V1,V2,V4}

CONSTRAINT PROPAGATION

- **Constraint propagation** **(filtering algorithm)** means the process of applying filtering to the constraints in the CSP instance at hand.
- As filtering one constraint can reduce the domains of its variables, it can trigger further reduction when filtering another constraint that also involves the reduced-domain variables.
- Thus a constraint can be filtered multiple times during the constraint propagation process.
- Usually the process is run to the end, meaning until no more reduction happens in filtering any of the constraints. Or if this takes too long, then until some resource usage limit is exceeded.

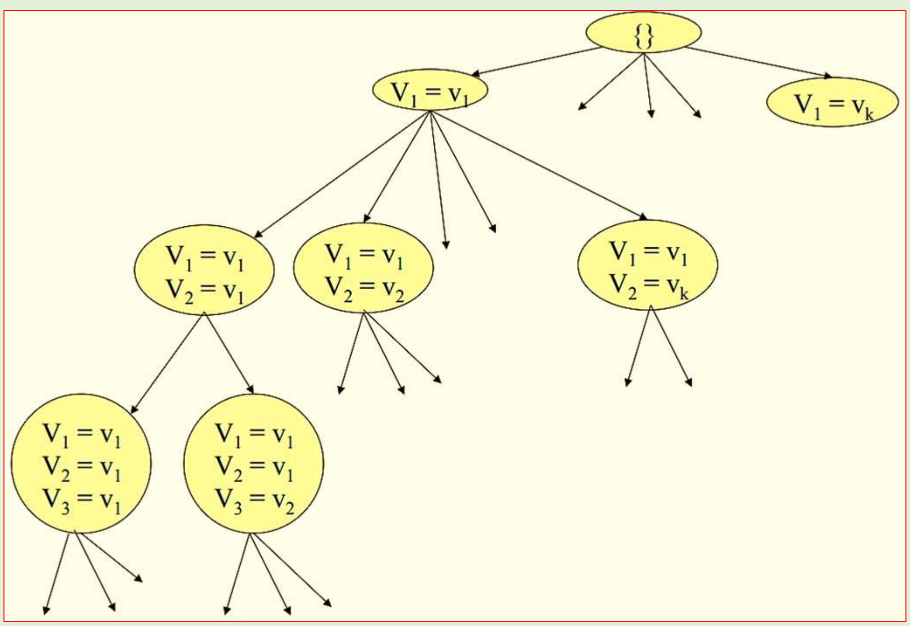
Example:

- When D_{before}(x)={1,2,4} and D_{before}(y)={1,2,5}, filtering the constraint x=y optimally gives to domains D_{after}(x)={1,2} and D_{after}(y)={1,2}. This is because the solutions of the constraint are x=1,y=1 and x=2,y=2.
- **constraint propagation**: Using the constraints to reduce the number of legal values for a variable, which in turn can reduce the legal values for another variable, and so on.

❖CSP as a Search Problem

- States: partial assignment of values to variables
- Start state: empty assignment
- Successor function: states with the next variable assigned
  - E.g., follow a total order of the variables $V_1, \ldots, V_n$
  - A state assigns values to the first k variables:
    - $\{V_1 = v_1, \ldots, V_k = v_k\}$
    - Neighbors of node $\{V_1 = v_1, \ldots, V_k = v_k\}$: nodes $\{V_1 = v_1, \ldots, V_k = v_k, V_{k+1} = x\}$ for each $x \in dom(V_{k+1})$
- Goal state: complete assignments of values to variables that satisfy all constraints
  - That is, models
- Solution: assignment (the path doesn't matter)

Prepared By Mr. EBIN PM, Chandigarh University, Punjab EDULINE 51



Prepared By Mr. EBIN PM, Chandigarh University, Punjab EDULINE 52

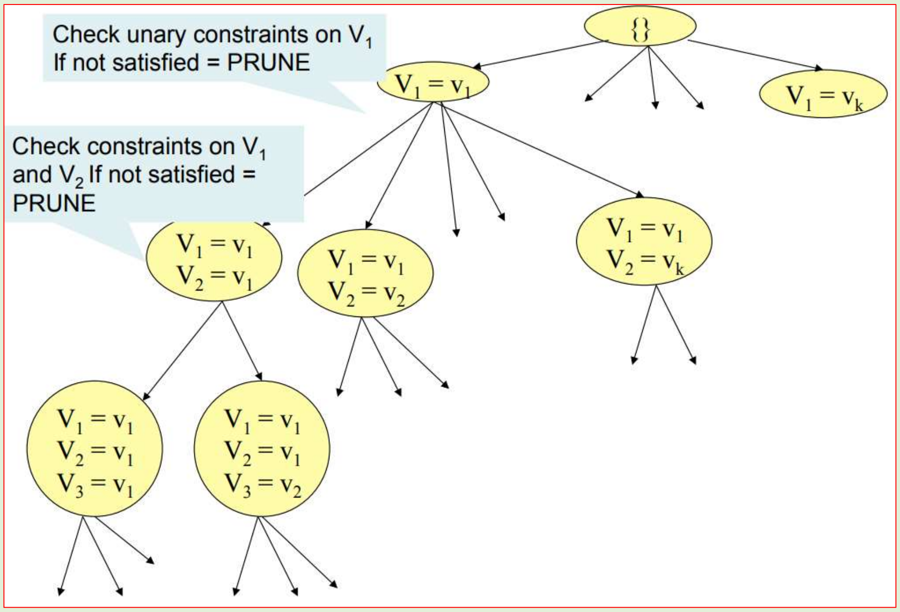
❖Backtracking algorithms

- Explore search space via DFS but evaluate each constraint as soon as all its variables are bound.
- Any partial assignment that doesn't satisfy the constraint can be pruned.
- A **partial assignment** is one that assigns values to only some of the variables

Example:

- 3 variables A, B, C each with domain {1,2,3,4}
- {A = 1, B = 1} is inconsistent with constraint A ≠ B regardless of the value of the other variables
  ⇒ Fail! Prune!

- Backtracking search, a form of depth-first search, is commonly used for solving CSPs.
- Backtracking algorithm repeatedly chooses an unassigned variable, and then tries all values in the domain of that variable in turn, trying to find a solution.
- If an inconsistency is detected, then BACKTRACK returns failure, causing the previous call to try another value.



- Performance heavily depends on the order in which variables are considered.
- Backtracking relies on one or more heuristics to select which variables to consider next.

➢ **E.g:** variable involved in the largest number of constraints: “If you are going to fail on this branch, fail early!”

➢ Can also be smart about which values to consider first