CONSTRAINT SATISFACTION PROBLEMS(CSP)

Introduction

- Constraint satisfaction problems are specific class of AI problems.
- CSPs involves set of variables each of which can take a domain of possible values and set of constraints to be satisfied by the variables.
- CSP is represented as a triple {X,D,C} where

  X={x1,x2,x3,..} where X is a set of variables

  D={D1,D2,D3,..} where each Di is the set of possible values of xi variable

  C={C1,C2,C3,….} where Ci is the constraint that restricts the values that can be assigned to subset of variables.

Definition

- **Variables:** Variables are used to represent entities or components of a problem for which values are to be assigned.

- Different variable types: Integer type, Boolean type, or Categorical type. The choice of variables depends on the problem being solved.
- Example: In scheduling problem time slots, tasks, resources, users are the variables.

- **Domains:** Domain specifies the range or set of values a variable can take.

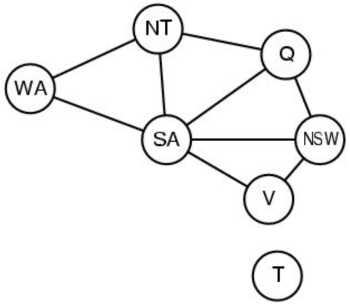
  Each variable in the CSP is associated with a domain.

  The domain restricts the values of the variables.

  The category of domain can vary depending on the nature of the problem being solved.

Domain Categories

- **Finite Domain**: In finite domains the variables can take only a set of discrete values.
- **Binary Domains:** Variable may consist of only two values {0,1}
- **Integer Domain:** Variable may consist of limited number of integer values like 1,2,3,4 etc.
- **Enumeration Domain consists** of limited number of distinct values like “red, green and blue”
- Ex: In graph coloring problem the domain is available colors.
- **Infinite domains** have an infinite number of possible values, such as real numbers.
- Ex: sine and cosine functions can take infinite domain values
- **Continuous Domain:** Some CSPs contain variables whose domains are continuous values and can accept any real number falls in a given range.
- **Real-valued domains:** Variables may accept any real number that falls within a given range like x[0,1] or x[1,100] etc..
- **Interval domains:** Variable may accept any real number that falls with in a given an interval domain like x϶[-Π , Π]
- Example: In scheduling the domain of task is available time slots
- Constraint graph: nodes are variables, arcs are constrai



Constraints

- **Constraints:** Constraints restrict the combinations of values that a variable can take.
- Constraints are represented in the form of logical expressions, equations or functions.
- Commonly used constraints are:

  Unary Constraints: These constraints limit the possible values of a single variable without considering the values of other variables. It has only one parameter. Ex : $x1 \neq 7$.

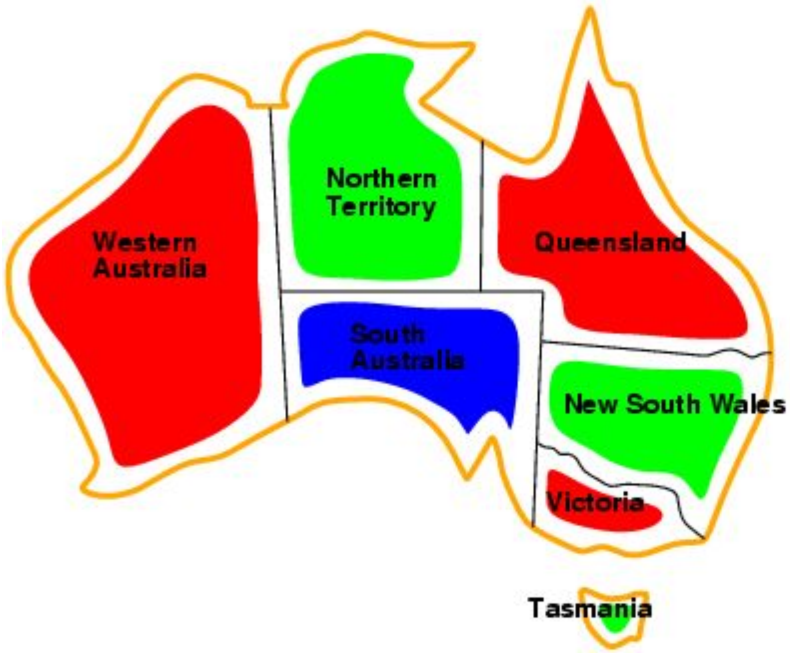
  Binary Constraints: Binary constraints describe the relationship between two variables. It has only two parameters. Ex: $x1 < x2$

  Global constraints: Global constraints involve multiple variables and restrict more complex relationship between the variables.
- Global constraint enforces all variables in a set must take distinct values.
- Ex: If two tasks are to be scheduled with one resource and two cannot be scheduled at a time then this is a global constraint.

Example: Map-Coloring

- Variables *WA, NT, Q, NSW, V, SA, T*
- Domains D_i = {red, green, blue}
- Constraints: adjacent regions must have different colors
  e.g., WA ≠ NT, or (WA, NT) in {(red, green),(red, blue),(green, red), (green, blue),(blue, red),(blue, green)}

Example: Map-Coloring



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

Soft Constraints

- These are the constraints which can be violated to a certain degree.
- Based on the degree of violation soft constraints assign penalties and cost to the problem solving.
- Solving the CSP problem with soft constraints involves optimization of the objective function to minimize the cost.
- **Example:** In project scheduling, meeting deadlines is a hard constraint, but it can be delayed slightly if the cost is reduced.

Representation of Constraint Satisfaction Problems (CSP)

- In **Constraint Satisfaction Problems (CSP)**, the solution process involves the interaction of variables, domains, and constraints.

- **Structured representation of CSP**

Finite Set of Variables ($V1,V2,\ldots,Vn$)**:**
The problem consists of a set of variables, each of which needs to be assigned a value that satisfies the given constraints.

Non-Empty Domain for Each Variable ($D1,D2,\ldots,Dn$)**:**
Each variable has a domain—a set of possible values that it can take. For example, in a Sudoku puzzle, the domain could be the numbers 1 to 9 for each cell.

Finite Set of Constraints ($C1,C2,\ldots,Cm$**):**
Constraints restrict the possible values that variables can take. Each constraint defines a rule or relationship between variables.

Constraint Representation:
Each constraint C_i is represented as a pair **<scope, relation>**, where:

- **Scope:** The set of variables involved in the constraint.
- **Relation:** A list of valid combinations of variable values that satisfy the constraint.

- Constraint satisfaction is a search procedure that operates in a space of constraint sets.
- The Initial state contains the constraints, given in the problem description.
- He goal state is any state that has been constrained *enough*, where *enough* must be defined for each problem.
- Constraint satisfaction is a two step process:

1. First, constrained are discovered and propagated as far as possible through out the system.
2. If there is still not a solution, search begins.

- A guess about something is made and added as a new constraint and so forth.
- Constraint propagation terminates when (i) a contradiction is detected, (ii) the propagation has run out of steam and no further changes can be made on the basis of the current knowledge.
- Constraints are two types: (i) holds list of possible values (ii) describes relationships between the objects.

Constraint Satisfaction Problem solving Techniques

1. Backtracking method
2. Constraint propagation method
3. Local Search method

Backtracking Method

- A variable is picked up, setting a value for it, and then recursively scanning through other variables.
- In the event of conflict, i.e. when it encounters constraints that cannot be satisfied it backtracks and tries a different value for preceding variable.
- Continue this process until all variables are assigned values, or a valid solution is found.
- The backtracking algorithm’s essential elements are:

Variable Ordering: The order in which variables are chosen is known as variable ordering.

Value Ordering: The sequence in which values are assigned to variables is known as value ordering.

Constraint Propagation: Reducing the domain of variables based on constraint compliance is known as constraint propagation.

Forward Checking

- The backtracking technique has been improved using forward checking.
- It tracks the remaining accurate values of the unassigned variables after each assignment and reduces the domains of variables whose values don't match the assigned ones.
- As a result, the search space is smaller, and constraint propagation is more effectively accomplished.
- Constraint Propagation is a process of communicating the domain reduction of a decision variable for all other constraints stated over this variable.
- It narrows down the domains of variables by iteratively applying constraints.
- It's often used in conjunction with backtracking to improve efficiency.

- **Step 1:** Start with an initial CSP problem in AI with variables, domains, and constraints.
- **Step 2:** Apply constraints that have been specified in the problem to narrow down the domains of variables.
- **Step 3:** After constraint propagation, some variables may have their domains reduced to only a few possibilities, making it easier to find valid assignments.
- **Step 4:** If a variable's domain becomes empty during propagation, it indicates that the current assignment is inconsistent, and backtracking is needed.

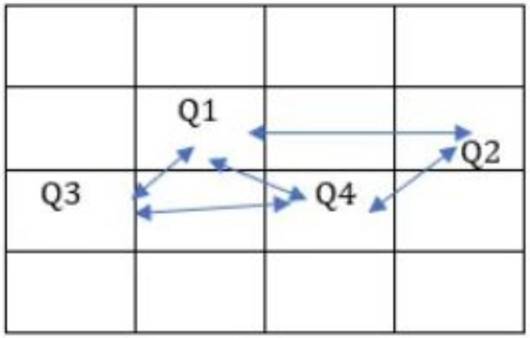
- **Example:**
- CSP with two variables, X and Y, each with domains {1, 2, 3}, and a constraint X ≠ Y.
- Constraint propagation will iteratively reduce the domains as follows: If X is assigned 1, then Y cannot be 1, so Y's domain becomes {2, 3}.

Local Search

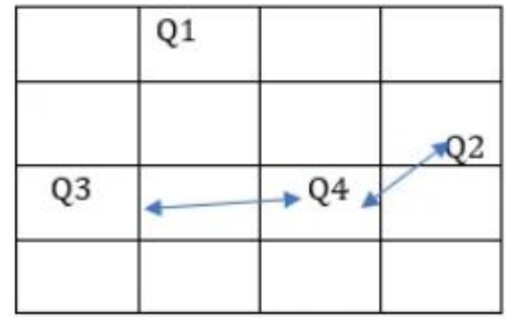
- Local search algorithms in AI are the methods used to find the solutions to optimization problems where the search space is continuous.
- The algorithm explores the neighborhood of the current solution to find a better solution and evaluates for the maximization of objective function and selects one of the neighborhoods and moves to that solution towards an optimal or near-optimal point.
- **N-Queens problem**: In this problem all the N-queens are to be placed on \
  N × N chess board.
- It can be represented as a CSP problem:
- N-Queens={Q,P,C}
- Where Q is a set of queens Q={q1,q2,q3,q4}
- P is set of positions for queens={{(1,1),(1,2),(1,3),(1,4)}, {(2,1),(2,2),(2,3),(2,4)}, {(3,1),(3,2),(3,3),(3,4)}, {(4,1),(4,2),(4,3),(4,4)}}
- C is a set of constraints for all the queens that no two queens should attack each other.
- C={No two queens should be in the same row, No two queens should be in the same column, No two queens should be in the diagonal}

4-Queens Problem

- Step1: Randomly select the solution state and check for constraints to validate whether the queens are correctly placed.
- In the initial state there are 5 pairs of queens attacking each other:



Initial state



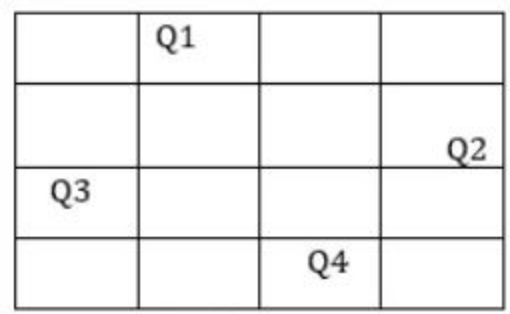
Intermediate state

- oring node to move to reduce the confl
- Q1 has 3 conflicts
- Q2 has 2 conflicts
- Q3 has 2 conflicts
- Q4 has 3 conflicts
- Among these queens we need to choose

the queen with maximum number of conflicts.

- So either Q1 or Q4 is to be moved.
- Let Q1 is selected to move.

- Q3 , Q4,Q2 all have one conflict. choose one among them to move.
- Let Q4 choose to move.



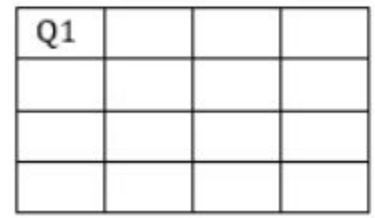
Final state

Backtracking and constraint propagation approach to solve N-Queens problem

- **Step1:** start with empty board and choose an unassigned variable. Q1
- · Select a value from its domain.
- P1 ={{(1,1),(1,2),(1,3),(1,4)}, {(2,1),(2,2),(2,3),(2,4)}, {(3,1),(3,2),(3,3),(3,4)}, {(4,1),(4,2),(4,3),(4,4)}}
- Choose one position (1,1) and place Q1
- Check if the assignment violates any constraints.
- Now propagate the remaining constraints to the next variable.
- Step2 : Choose another variable Q2
- The domain for Q2 is P2. After adding some more constraints

search space reduces and the domain becomes

P2= {{(2,3),(2,4)}, {(3,2),(3,4)}, {(4,2),(4,3)}}

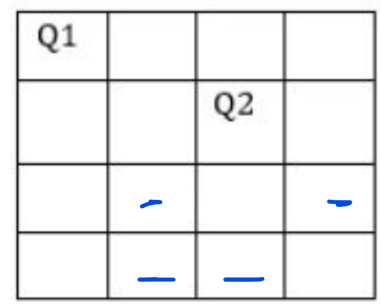


initial state

- Constraints C2 for Q2= {cannot place queen in first row, cannot place queen

in second row, cannot place 2 queens in same row or same column or diagonal}

- Now choose first free place (2,3) and check whether there is any conflict
- It is not same row or column with Q1. Choose that place.



- Step 3: Choose another variable Q3
- P3={{(3,2),(3,4)}, {(4,2),(4,3)}}
- Constraints C3 for Q3= {cannot place queen in first row, cannot place queen in second row, cannot place 2 queens in same row or same column or diagonal, cannot place queen in 1st or 2nd row, cannot place queen in 1st or 2nd column}
- so the domain for Q3 reduces to P={{(3,4)},{(4,3)}}
- choose first free position and place Q3
- If a constraint is violated, backtrack

to the previous variable, and try another value.

- Now constraint is violated with Q3 and Q2

are becoming diagonal.

Q1			
		Q2	
			Q3

- Now try to place the Q3 in (4,3)
- This placement is also violating the constraint that

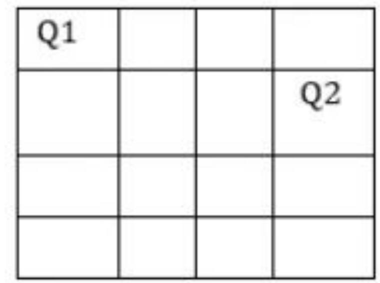
no two queens can be placed in same column.

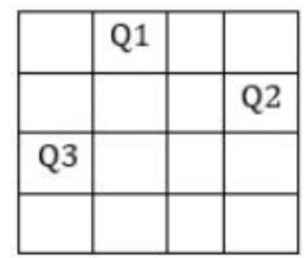
- And there is no other place for Q3 . i.e the domain

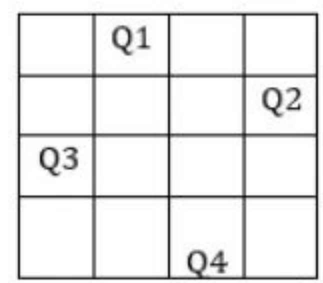
is exhausted.

Q1			
		Q2	
		Q3	

- Now backtrack to the previous variable i.e Q2 and try another value in its domain. The domain of Q2 is P2.
- P2={{(2,3),(2,4)}, {(3,2),(3,4)}, {(4,2),(4,3)}} initially we have chosen (2,3).
- Now choose the next value(2,4)
- Now the domain for Q3 changes to P3={{ {(3,2)}, {(4,2)}}
- Choose first position and place Q3 in (3,2)
- Now the domain for Q4 becomes only (4,3) which is also a conflict position.
- Now we need to back track to Q1 and find the another position for Q1 whose domain is P1=={{(1,1),(1,2),(1,3),(1,4)}, {(2,1),(2,2),(2,3),(2,4)}, {(3,1),(3,2),(3,3),(3,4)}, {(4,1),(4,2),(4,3),(4,4)}}
- Choose (1,2) for Q1
- Choose place for Q2 in {(2,4)}
- Continue this process until all variables are assigned values, or a valid solution is found.
- Choose place for Q3 in {(3,1)}
- Choose place for Q4 in {(4,3)} which is a final solution.
- The solution for the Constraint Satisfaction Problems can be solved efficiently using backtracking algorithm in combination with constraint propagation method.



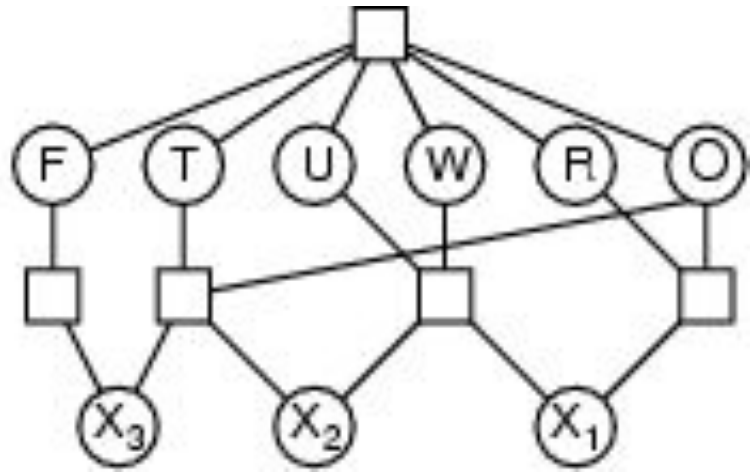




4-Queens problem solution

Example: Cryptarithmetic

```
    T W O
+   T W O
---------
  F O U R
```



- Variables: *F T U W R O* X_1 X_2 X_3
- Domains: {*0,1,2,3,4,5,6,7,8,9*}
- Constraints: *Alldiff (F,T,U,W,R,O)*

  - $O + O = R + 10 \cdot X_1$
  - $X_1 + W + W = U + 10 \cdot X_2$
  - $X_2 + T + T = O + 10 \cdot X_3$
  - $X_3 = F$, $T \neq 0$, $F \neq 0$

Challenges in Solving CSPs

- While CSPs offer a powerful framework for solving many AI problems, they also pose several challenges:

- **Scalability**: As the number of variables and constraints increases, the size of the solution space grows exponentially, making it challenging to find solutions in a reasonable time.

- **Dynamic CSPs**: In real-world problems, constraints and variables may change over time, making it necessary to adapt the solution dynamically. These are known as **Dynamic CSPs**, and solving them efficiently requires specialized techniques.
- **Inconsistent or Over-constrained Problems**: Sometimes, it is impossible to satisfy all constraints, leading to **inconsistent** problems.

- Techniques like constraint relaxation or optimization approaches can help in finding acceptable solutions in such cases.