



CS-208:Artificial Intelligence

Topic-19: Constraint Satisfaction Algorithm

Constraint Satisfaction

Many Problems in AI can be viewed as problems of constraint satisfaction in which the goal is to discover some problem state that satisfies a given set of constraints.

Examples:

- Crypt arithmetic Problem
- Map Colouring Problem
- Some real world problem perceptual labeling problem
- Many design task in which design must be created within fixed limits on time, cost and materials
- etc

Constraint Satisfaction Problems

- In general a CSP is a problem composed of a finite set of variables each of which has a finite domain of values and a set of constraints. Each constraint is defined over some subset of the original set of variables and restricts the values these variables can simultaneously take. The task is to find an assignment of a value for each variable such that the assignments satisfy all the constraints. In some problems the goal is to find all such assignments.
- A CSP is called an n -ary CSP when n is the maximum number of distinct variables over which a constraint is specified. For instance a binary CSP has constraints involving two variables only. It is possible that any n -ary CSP can be converted to a binary CSP.

Formal Definition

Formally a binary CSP can be defined to consist of a triple (V, D, R) , where

- V is a set $\{V_1, \dots, V_i, \dots, V_n\}$ of n variables;
- D is a set $\{D_1, \dots, D_i, \dots, D_n\}$ of domains such that $\forall i \ 1 \leq i \leq n$ and $D_i = \{v_1^i, \dots, v_j^i, \dots, v_m^i\}$ is the finite set of possible values for V_i .
- R is a sequence $\{\dots, R_{ij}, \dots\}$ of binary constraint relations such that $\forall R_{ij} \in R$, where R_{ij} constrains the two variables V_i and V_j and is defined by a subset of the Cartesian product $D_i \times D_j$. The set of pairs of values in R specifies the allowed pairs of values for variables V_i and V_j . If $(v_l^i, v_m^j) \in R_{ij}$ we say that the assignment $\{V_i \leftarrow v_l^i, V_j \leftarrow v_m^j\}$ is consistent.

A solution to CSP is a complete assignment that satisfies all the constraints

Defining of CSP in AI Context





For solving Constraint Satisfaction Problem

- The state is defined by variables V_i with values from domain D_i
- The goal test is a set of constraints specifying allowable combinations of values for subsets of variables.

Example: N-Queens

- **Problem:** *The N-queens problem can be modeled as a CSP. Given an integer N , the problem is to place N queens on N distinct squares in an $N \times N$ chess board satisfying the constraint that no two queens should threaten each other. Two queens threaten each other if and only if they are on the same row, column or diagonal.*

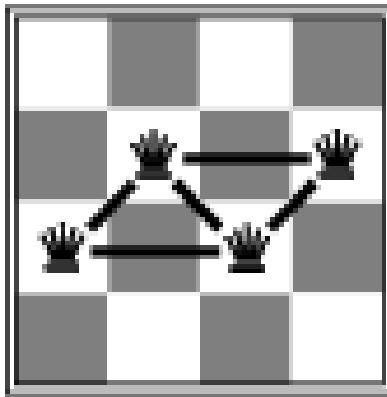
A possible Solution to 4-queens problem

	1	2	3	4
V_1				
V_2				
V_3				
V_4				

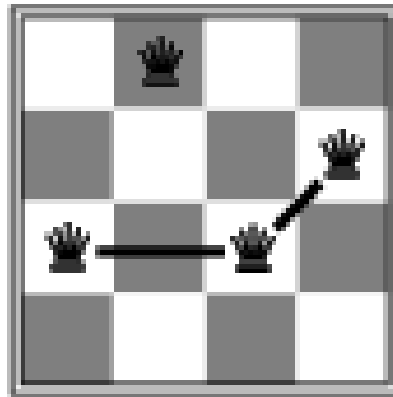
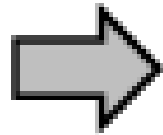
Illustrative Example: 4-Queens

Problem: Place the four queens on a 4×4 grid in such a way that no two queens are on the same diagonal, row or column.

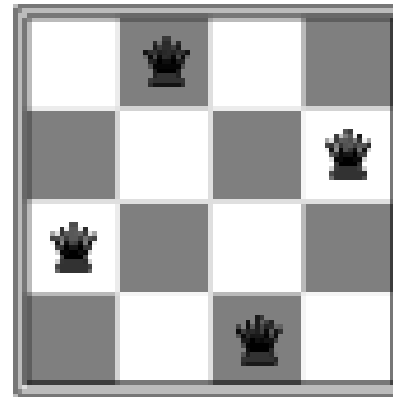
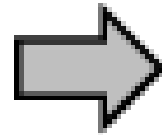
- **States:** 4 queens in 4 columns ($4^4 = 256$ possible states)
- **Actions:** Move queen in column
- **Goal test:** No attacks
- **Evaluation:** $h(n) =$ number of attacks



$h = 5$



$h = 2$



$h = 0$

Encoding of 4-queens problem as a CSP

- Make each row a variable $V = \{V_1, V_2, V_3, V_4\}$. The value of each variable will represent column in which the queen is row _{i} ($1 \leq i \leq 4$) is placed;
- Domain: $D = \{D_1, D_2, D_3, D_4\}$. Each of the four variables can take one the four columns as its value with label 1 to 4 . Therefore the domains of the four variables are $D_1 = D_2 = D_3 = D_4 = \{1, 2, 3, 4\}$;
- Constraints: $R = \{R_{ij} \mid i < j \text{ and } (1 \leq i \leq 4)\}$. For each constraint R_{ij}
- No two queens on the same row: This constraints become trivial given the variable encoding.
- No two queens on the same column: $V_i \neq V_j$
- No two queens on the same diagonal: $|i - j| \neq |V_i - V_j|$

Standard Approaches for CSP Solving

There are three standard approaches for solving CSP

- **Tree Search:** It is a Standard technique for solving CSPs. The basic algorithm is simple backtracking(BT), a general strategy which has been widely used in problem solving. It also serves as the basis for many other algorithm.
- **Constraint Propagation:** It is aimed at transforming a CSP into an equivalent problem that is easier to solve. Constraint propagation works by reducing the domain size of the variable in such a manner that no solution are ruled out. It involves removing redundant values from the domain of the variable and propagating the effect of these removal throughout the constraint.
- **Combined Approach of Backtracking and Constraint Propagation.**

Backtracking (BT) Algorithm

- In BT variables are instantiated one by one.
- When a variable is instantiated a value from its domain is picked and assigned to it. Then constraint checks are performed to make sure that the new instantiation is compatible with all the instantiations made so far.
 - If all the completed constraints are satisfied, this variable has been instantiated successfully and we can move on to instantiate the next variable.
 - If it violates certain constraints then an alternative value when available is picked.
- If all the variables have a value, noting that all the assignments are consistent, then the problem is solved.
- If at any stage no value can be assigned to a variable without violating a constraint, backtracking occurs. That means the most recent instantiated variable has its instantiation revised and a new value if available is assigned to it. At this point we continue on to try instantiating the other variables or we backtrack farther. This carries on until either a solution is found or all the combinations of instantiation have been tried and have failed which means that there is no solution.

function BACKTRACKING-SEARCH(*csp*) **returns** a solution, or failure

return RECURSIVE-BACKTRACKING($\{\}$, *csp*)

function RECURSIVE-BACKTRACKING(*assignment*, *csp*) **returns** a solution, or failure

if *assignment* **is complete** **then return** *assignment*

var \leftarrow SELECT-UNASSIGNED-VARIABLE(*Variables*[*csp*], *assignment*, *csp*)

for each *value* **in** ORDER-DOMAIN-VALUES(*var*, *assignment*, *csp*) **do**

if *value* **is consistent with** *assignment* **according to** Constraints[*csp*] **then**

add { *var* = *value* } **to** *assignment*

result \leftarrow RECURSIVE-BACKTRACKING(*assignment*, *csp*)

if *result* \neq failure **then return** *result*

remove { *var* = *value* } **from** *assignment*

return failure

Pseudo Code for BT Algorithm-PART 1

```
function    CONSISTENT( $V_i, v_l^i$ )  
for each( $V_j, v_m^j$ )  $\in$  solution  
    if  $R_{ij} \in R$  and  $(v_l^i, v_m^j) \notin R_{ij}$   
        return FALSE  
return TRUE
```

Pseudo Code for BT Algorithm-PART 2

```
function Search_BT( $Vars, Level$ )  
  select a variable  $V_i \in Vars$   
  for each value  $v_l^i \in D_i$   
    if CONSISTENT( $V_i, v_l^i$ )  
       $Solution \leftarrow Solution + (V_i, v_l^i)$   
      if  $V_i$  is the only variable in  $Vars$   
        return TRUE  
      else  
        if SEARCH_BT( $Vars \setminus \{V_i, Level + 1\}$ )  
          return TRUE  
  return FALSE
```

Pseudo Code for BT Algorithm-PART 3

function BT

Solution $\leftarrow 0$

return SEARCH_BT(*v*,1)

Search Tree of the 4-queens problem using BT

