Chapter 2: Constraint Satisfaction Problems (CSP)

CONSTRAINT PROGRAMMING COURSE

M1 IDO

Definition

A Constraint Satisfaction Problem (CSP) is a problem modelled as a set of constraints imposed on variables taking values in specified domains. Formally, a CSP is a triple (X, D, C) where:

- $X = \{X_1, ..., X_n\}$: a set of variables.
- $D = \{D_1, ..., D_n\}$: a set of domains, with $X_i \in D_i$ for $1 \leq i \leq n$.
- $C = \{C_1, ..., C_m\}$: a set of constraints.

Binary Relations Associated with a Constraint

Let P = (X, D, C) be a binary discrete CSP, c_k a constraint of P, and X_i, X_j the variables on which c_k bears. The binary relations associated with c_k are:

- $Rk(X_i, X_j) = \{(a,b) \in D(X_i) \times D(X_j) : (X_i, X_j) = (a,b) \text{ satisfies } c_k\}$
- $Rk(X_j, X_i) = \{(a,b) \in D(X_j) \times D(X_i) : (X_j, X_i) = (a,b) \text{ satisfies } c_k\}$

Worked Example: Given

- X = {X1, X2, X3, X4}
- D = {D1, D2, D3, D4} with D1 = D2 = D3 = D4 = {0, 1}
- C = {X1 ≠ X2, X3 ≠ X4, X1 + X3 < X2}

R_1	
X_1	X_2
0	1
1	0

R_2	
X_3	X_4
0	1
1	0

R_3		
X_1	X_2	X_3
0	1	0

The relations R1 (for X1 ≠ X2) and R2 (for X3 ≠ X4) each contain the pairs {(0,1), (1,0)}. The relation R3 (for X1 + X3 < X2) contains the single tuple {(0,1,0)}.

Modelling Problems as CSPs

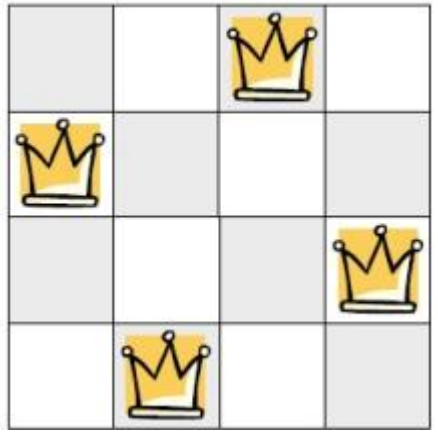
General methodology:

- Identify the variables (the unknowns).
- Identify the domains of those variables.
- Identify the constraints.

Multiple models are often possible. Choice criteria include *simplicity of expression* and *efficiency* (size of the search space).

Example: The 4-Queens Problem

Place 4 queens on a 4×4 board so that no two queens are in the same row, column, or diagonal.
(X = ?, D = ?, C = ?)



$X = \{X_1, X_2, X_3, X_4\}$ (column position of queen in row $_i$)
$D = \{D_1, D_2, D_3, D_4\}$ with $D_i = \{1, 2, 3, 4\}$
C = {C_LI, C_DA, C_DD} where:

$C\_LI = \{X_i \neq X_j : i \neq j\}$ (no two queens in the same column)

$= \{X_1 \neq X_2 , X_1 \neq X_3 , X_1 \neq X_4 , X_2 \neq X_3 , X_2 \neq X_4 , X_3 \neq X_4 \}$

$C\_DA = \{X_i + i \neq X_j + j : i \neq j\}$ (no two queens on the same ascending diagonal)

$= \{X_1 +1 \neq X_2 +2 , X_1 +1 \neq X_3 +3, X_1 +1 \neq X_4 +4 , X_2 +2 \neq X_3 +3 , X_2 +2 \neq X_4 +4 , X_3 +3 \neq X_4 +4\}$

$C\_DD = \{X_i - i \neq X_j - j : i \neq j\}$ (no two queens on the same descending diagonal)

$= \{X_1 -1 \neq X_2 -2 , X_1 -1 \neq X_3 -3, X_1 -1 \neq X_4 -4 , X_2 -2 \neq X_3 -3 , X_2 -2 \neq X_4 -4 , X_3 -3 \neq X_4 -4\}$

Resolution Approach

Naïve approach: try all possible assignments. This is not feasible for real-valued or integer domains. For finite domains, smarter strategies are needed. One may encounter fundamental obstacles:

- Undecidability for some classes of constraints.
- NP-completeness (or worse) in general.

One may search for:

- Any one solution.
- All solutions.
- An optimal or near-optimal solution according to a cost or objective function – this is a Constraint Optimisation Problem (COP).

Complexity: CSPs are NP-hard in general.

Solving a CSP

Solving a CSP means finding an assignment of values to variables such that all constraints are satisfied.

- **Assignment:** an instantiation of variables over their domains.
- **Total assignment**: all variables are assigned.
- **Partial assignment**: only some variables are assigned.
- **Consistent assignment**: no constraint is violated.
- **Inconsistent assignment**: at least one constraint is violated.
- **Solution**: a total and consistent assignment.

Example

X = {X1, X2, X3, X4}

D = {D1, D2, D3, D4} avec D1 = D2 = D3 = D4 = {0, 1}

C = {X1 ≠ X2, X3 ≠ X4, X1 + X3 < X2}

A = {(X2, 0), (X3, 1)} **Assignment**

A = {(X1, 0), (X2, 0), (X3, 0), (X4, 0)} **total assignment**

A = {(X1, 0), (X2, 0)} **partial assignment**

A = {(X1, 0), (X2, 0)} **inconsistent assignment**

A = {(X3, 0), (X4, 1)} **consistent assignment**

A = {(X1, 0), (X2, 1), (X3, 0), (X4, 1)} **solution**

Binarisation of Constraints

Although constraints can be n-ary, there are two special cases of interest:

- **Unary constraints**: handled by preprocessing on the domains.
- **Binary constraints**: if all constraints are binary, the CSP can be represented as a constraint graph whose vertices are the variables and whose edges are the constraints.

Any n-ary constraint can be expressed in terms of binary constraints; hence any CSP can be represented as a binary CSP.

Binarisation of Constraints
Method 1: Preserving Original Variables

Create an encapsulated variable U whose domain is the Cartesian product of the encapsulated variables. Apply the original n-ary constraint to reduce the domain, then add binary constraints linking each original variable to U via a projection: $X_i = pos_i(U)$.

Example: $X = \{X_1, X_2, X_3\}$, $D_1=\{1,2\}$, $D_2=\{3,4\}$, $D_3=\{5,6\}$, $C = \{X_1+X_2=X_3, X_1<X_2\}$.

Binary CSP : $X' = \{X_1, X_2, X_3, U\}$, $D' = \{D_1, D_2, D_3, D'_U\}$
Create U with domain $D_U = D_1 \times D_2 \times D_3$ and apply $X_1+X_2=X_3$ to get $D'_U = \{(1,4,5),(2,3,5),(2,4,6)\}$.

New constraints: $C' = \{X_1<X_2, X_1=pos_1(U), X_2=pos_2(U), X_3=pos_3(U)\}$.

Binarisation of Constraints
Method 2: Without Preserving Original Variables

Create encapsulated variables U and V, one per original n-ary constraint. Each encapsulated variable has domain equal to the Cartesian product of the variables it encapsulates. Reduce domains by applying the constraints, then link shared positions across encapsulated variables: $pos_i(U) = pos_j(V)$.

Example: same problem as above.

Binary CSP : $X' = \{U, V\}$, $D' = \{D'U , D'V\}$, C'

$U = \{(X_1,X_2,X_3) : X_1+X_2=X_3\}$, $D'_U = \{(1,4,5),(2,3,5),(2,4,6)\}$.

$V = \{(X_1,X_2) : X_1<X_2\}$, $D'_V = \{(1,3),(1,4),(2,3),(2,4)\}$.

New constraints: $C' = \{X_1=pos_1(U), X_2=pos_2(U), X_3=pos_3(U), X_1=pos_1(V), X_2=pos_2(V)\}$.