Constraint Satisfaction Problem

A solution to CSP is assignment to each variable such that each constraint is satisfied.

Ex: n-Queens Problem

Given $\boxed{\mathbb{R} = \langle X, D, C \rangle}$

Sol($\mathbb{R}$) (scope, relat

For a binary CSP

An assignment $\bar{a}$ on scope S $\{x_1, x_2, \ldots\}$ satisfies a constraint C_i, iff $S_i \subseteq S$ and $\pi_{S_i} \bar{a} \in R_i$

$S_i \subseteq S \{Q_1, Q_2, Q_3, Q_4\}$.

$R_{12} = \{Q_1, Q_2\}$

$a =$

$\mathbb{R} \rightarrow \{3, 1, 2, 4\}$

$R_{12} = \{(3,1)$

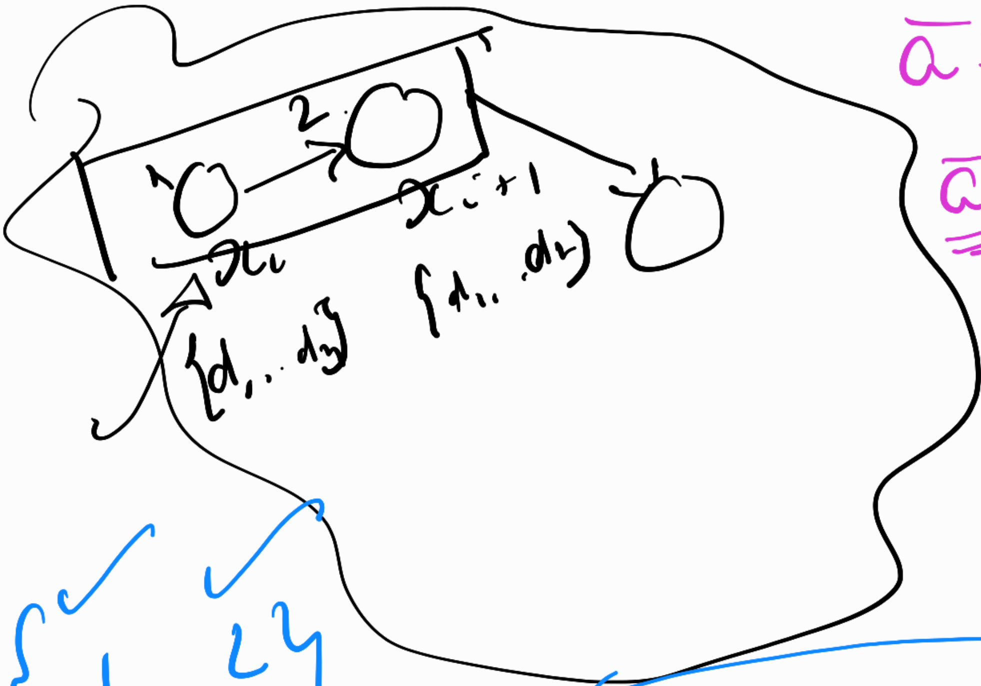
$\bar{a} = \{3, 1\}$

$\bar{a} = \{3, 1, 2\}$

R_{12} $S_i = \{Q_1, Q_2, Q_3\}$

R_{13}

R_{23}



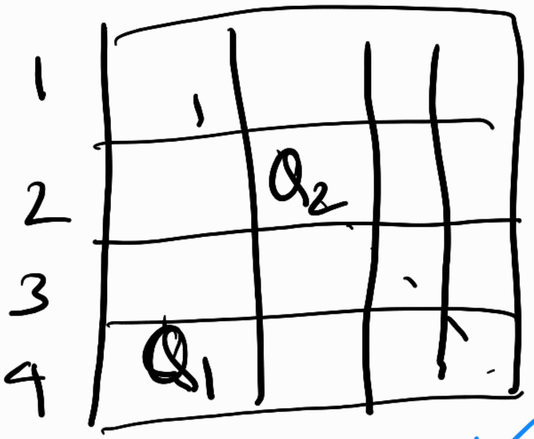
$\{1, 2\}$

$\{1, 3, 2\}$

$\bar{a} =$ Subset $\bar{a}$

$(k-1)^n$

k



Q_1, Q_2, Q_3, Q_4

$X \rightarrow$

4-Queens problem

$R_{12} = \{(1,3), (1,4), (3,1), (4,1), (4,2), (2,4)\}$, $R_{23}, R_{24}, R_{13}, R_{14}, R_{34}$

Binary CSP

$C\langle \underline{Scope}, Rela \rangle$.

$S_i \langle (\underline{Q_1}, \underline{Q_2}) \subseteq S, R_{12} \rangle$

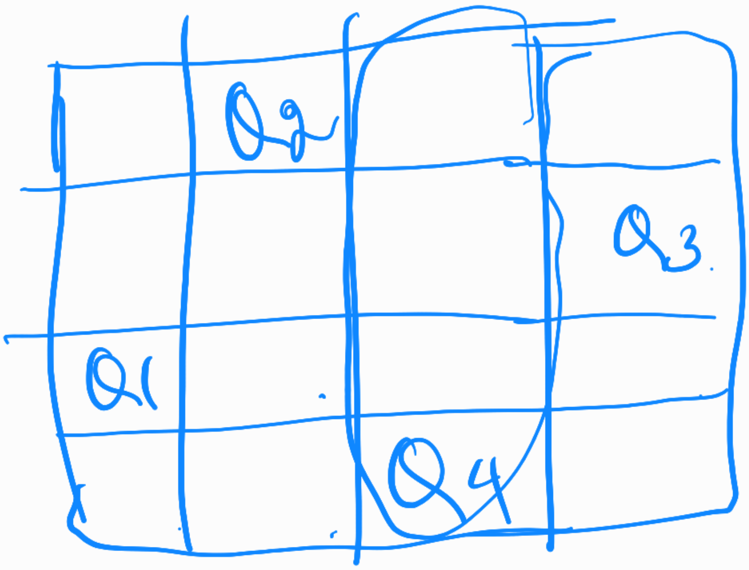
$\langle (Q_3, Q_4) \subseteq S, R_{34} \rangle$

Con

(Q_1, Q_3)

$R_{13} = \{ \langle 1,4 \rangle \}$

$R_{23} \{ (1,4) \}$



$\rightarrow \bar{a} = \{1, 3, 2\} \rightarrow$ Partial Solution

$\rightarrow a = \{3, 1, 2, 4\}$ satisfies constraint with scope S_i

4-Queens problem

$$R_{34} = \{ \circ, \circ \circ, \circ \}$$

$$R\{X, D, C\} \longrightarrow Sol(R)$$

$$\underline{R\{X, D, C\}} = \boxed{\{ (R_{12}), (R_{13}), (R_{14}), (R_{23}), (R_{24}), (R_{34}) \}}$$

$$Sol(R) \downarrow \; a \in$$

$$R_i \to \{a\}$$

$$R_{12} = \{ R_1, R_2 \}$$

$$R_{23} = \{ \underbrace{R_{12}}_{R_1}, \underbrace{R_{13}}_{R_2}, \underbrace{R_{23}}_{R_3} \}$$

Backtracking$(\cancel{X}, D, C)$:

$$i \leftarrow 1$$

$$D_i' \leftarrow D_i$$

$$a = ()$$

While $1 \le i \le n$.

$\underline{\underline{x}} \leftarrow$ select (X_i, D_i', c).

If $x == \text{NULL}$.

$i \leftarrow i-1$

else.

$i \leftarrow i+1$

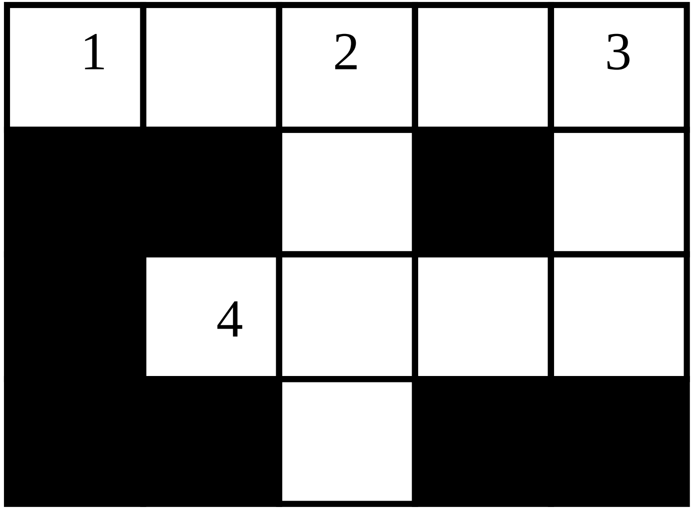
$D_i' \leftarrow D_i$

$a = (a, x)$;

CSP Solution Overview

- CSP Graph Creation:
  - Create a Node for Every Variable. All possible Domain Values are initially Assigned to the Variable
  - Draw edges between Nodes if there is a Binary Constraint. Otherwise Draw a hyper-edge between nodes with constraints involving more than two variables
- Constraint Propagation:
  - Reduce the Valid Domains of Each Variable by Applying Node Consistency, Arc / Edge Consistency, K-Consistency, till no further reduction is possible. If a solution is found or the problem found to have no consistent solution, then terminate
- Search for Solution:
  - Apply Search Algorithms to Find Solutions
  - There are interesting properties of CSP graphs which lead of efficient algorithms in some cases: Trees, Perfect Graphs, Interval Graphs, etc
  - Issues for Search: Backtracking Scheme, Ordering of Children, Forward Checking (Look-Ahead) using Dynamic Constraint Propagation
  - Solving by Converting to Satisfiability (SAT) problems

CSP Graph for Crossword



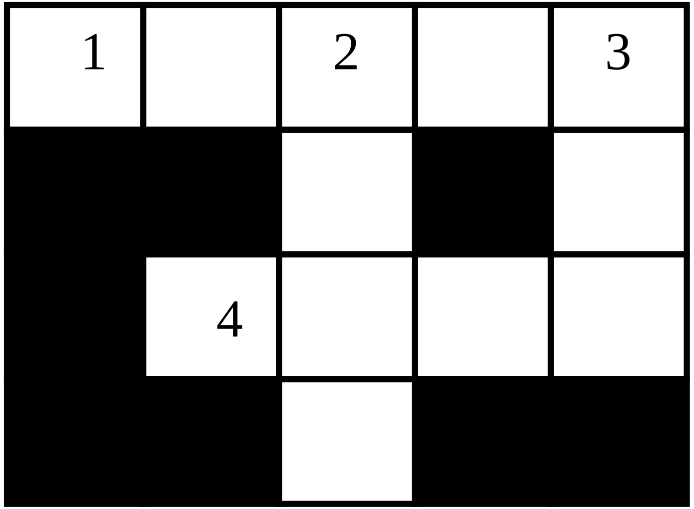
Word List:

astar, happy, hello, hoses, live, load, loom, peal, peel, save, talk, ant, oak, old

Constraint Propagation Steps

- Constraints
  - Unary Constraints or Node Constraints
  - Binary Constraints or Edges between CSP Nodes
  - Higher order or Hyper-Edges between CSP Nodes
- Node Consistency
  - For every Variable V_i, remove all elements of D_i that do not satisfy the Unary Constraints for the Variable
  - First Step is to reduce the domains using Node Consistency
- Arc Consistency
  - For every element x_ij of D_i, for every edge from V_i to V_j, remove x_ij if it has no consistent value(s) in other domains satisfying the Constraints
  - Continue to iterate using Arc Consistency till no further reduction happens.
- K-Consistency or Path Consistency
  - For every element y_ij of D_i, choose a Path of length L with L variables, use a consistency checking method similar to above to reduce domains if possible

CSP Graph for Crossword



Word List:

astar, happy, hello, hoses, live, load, loom, peal, peel, save, talk, ant, oak, old

Applying Node Consistency:

D1 = {astar, happy, hello, hoses}
D2 = {live, load, loom, peal, peel, save, talk}
D3 = {ant, oak, old}
D4 = {live, load, loom, peal, peel, save, talk}

NOW APPLY ARC CONSISTENCY

Applying Arc Consistency:

D1 = {astar, happy, hello, hoses}

D2 = {live, load, loom, peal, peel, save, talk}

D3 = {ant, oak, old}

D4 = {live, load, loom, peal, peel, save, talk}

Arc Consistency Algorithm AC-3

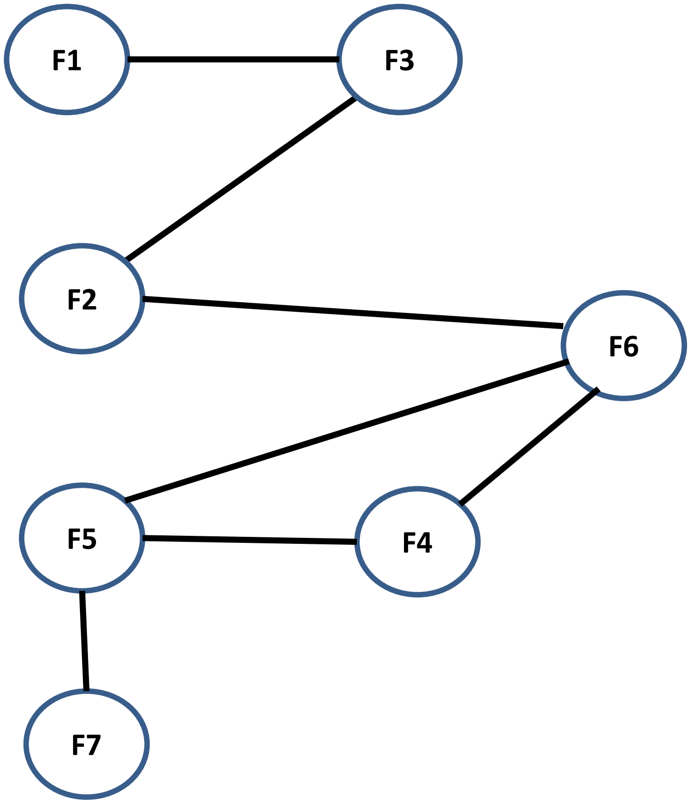
```
AC-3(csp) // inputs - CSP with variables, domains, constraints
    queue ← local variable initialized to all arcs in csp
    while queue is not empty do
        (Xi, Xj) ← pop(queue)
        if Revise(csp, Xi, Xj) then
            if size of Di = 0 then return false
            for each Xk in Xi.neighbors-{Xj} do
                add (Xk, Xi) to queue
    return true

Revise(csp, Xi, Xj)
    revised ← false
    for each x in Di do
        if no value y in Dj allows (x, y) to satisfy constraint between Xi and Xj then
            delete x from Di
            revised ← true
    return revised
```

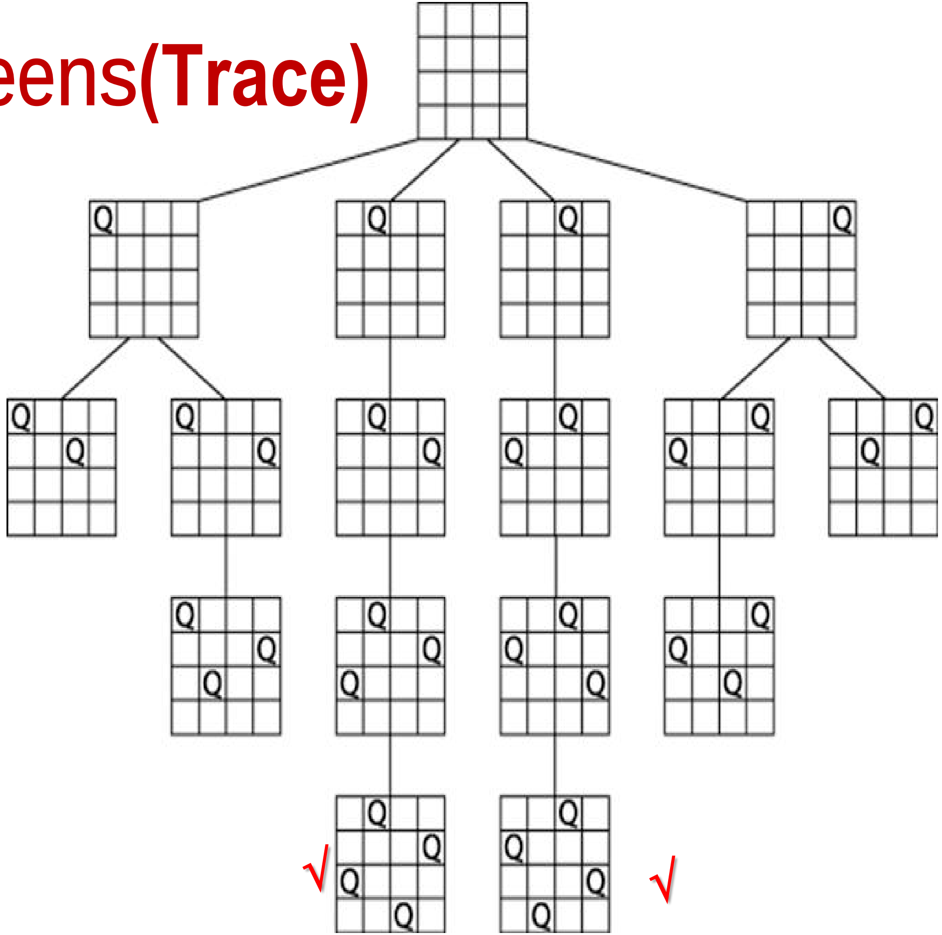
Time complexity: $O(cd^3)$

Backtracking for Airline Gate Scheduling

Flight No	Dep Time	G Start	G End
F1	7:00	6:15	7:15
F2	8:30	7:45	8:45
F3	7:45	7:00	8:00
F4	9:45	9:00	10:00
F5	10:00	9:15	10:15
F6	9:00	8:15	9:15
F7	11:00	10:15	11:15



Search 4-Queens(**Trace**)



Strategies for CSP Search Algorithms

- Initial Constraint Propagation
- Backtracking Search
  - Variable Ordering
    - Most Constrained Variable / Minimum Remaining Values
    - Most Constraining Variable
  - Value Ordering
    - Least Constraining Value leaving maximum flexibility
  - Dynamic Constraint Propagation Through Forward Checking
    - Preventing useless Search ahead
  - Dependency Directed Backtracking
- SAT Formulations and Solvers
- Optimization
  - Branch-and-Bound
  - SMT Solvers, Constraint Programming
- Learning, Memoizing, etc
- CSP Problems are NP-Hard in General