SEARCH IN AI

CONSTRAINT SATISFACTION PROBLEMS (CSP)

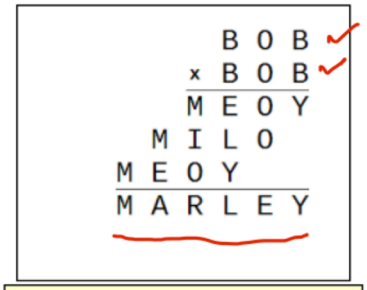
given a configuration

valid configuration

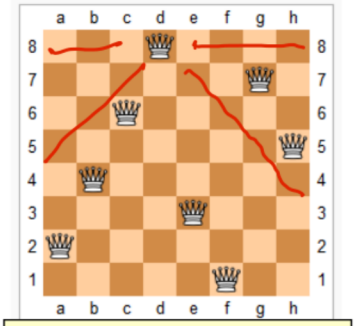
solve a set of constraints

Partha P Chakrabarti
Indian Institute of Technology Kharagpur

Constraint Satisfaction Problems (CSPs)

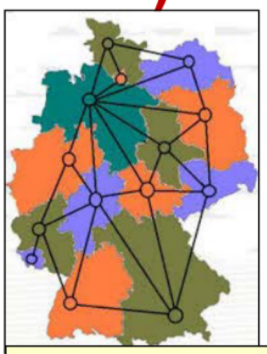


CRYPTARITHMETIC PUZZLE

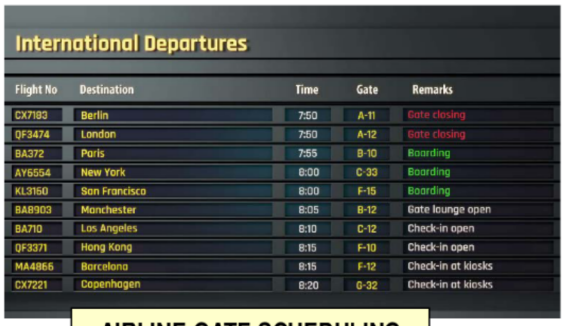


N-QUEENs

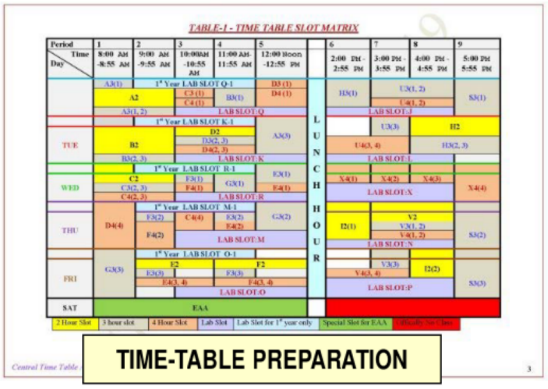
CROSSWORD PUZZLE



MAP COLOURING



AIRLINE GATE SCHEDULING



TIME-TABLE PREPARATION

KNAPSACK

Basic CSP Formulation

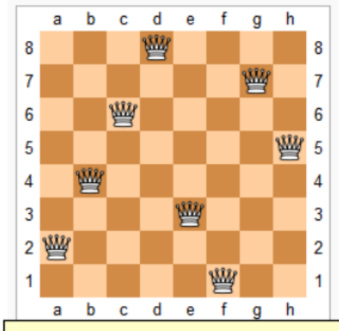
- **Variables**
  - A Finite Set of Variables V_1, V_2, …., V_n
- **Domains**
  - Each Variable has a Domain D_1, D_2, …., D_n from which it can take a value.
  - The Domains may be discrete or continuous domains
- **Satisfaction Constraints**
  - A Finite Set of Satisfaction Constraints, C_1, C_2, …C_m
  - Constraints may be unary, binary or be among many variables of the domain
  - All Constraints have a Yes / No Answer for Satisfaction given values of variables
- **Optimization Criteria (Optional)**
  - A Set of Optimization Functions O_1, O_2, ….O_p
  - These Optimization Functions are typically max or min type
- **Solution**
  - To Find a Consistent Assignment of Domain Values to each Variable so that All Constraints are Satisfied and the Optimization Criteria (if any) are met.

Formulating CSPs

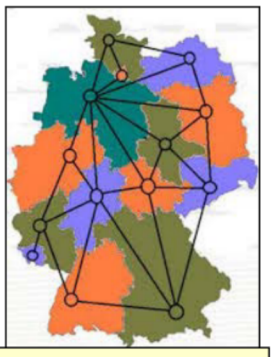
1. VARIABLES
2. DOMAINS
3. SATISFACTION CONSTRAINTS
4. OPTIMIZATION CRITERIA
5. SOLUTION

```
      B O B
    x B O B
    M E O Y
  M I L O
M E O Y
M A R L E Y
```

CRYPTARITHMETIC PUZZLE



N-QUEENs



MAP COLOURING

B, O, M, E, Y, I, L, R

Variables

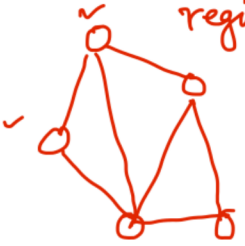
Domain : {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Constraints :- Uniqueness
Multiplication operator

row is a variable

D : a, b, ..., h

constraints



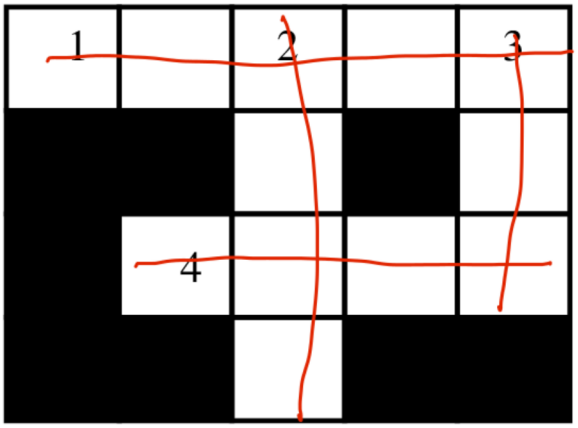
region : variable

{1, 2, 3}

Domain

min no. of colours

Formulating CSPs: Crossword



Word List:

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

1. VARIABLES
2. DOMAINS
3. SATISFACTION CONSTRAINTS
4. OPTIMIZATION CRITERIA
5. SOLUTION

variables: 1, 2, 3, 4

Domain: Word list

Constraints → (1,2), (1,3), (2,4), (3,4)

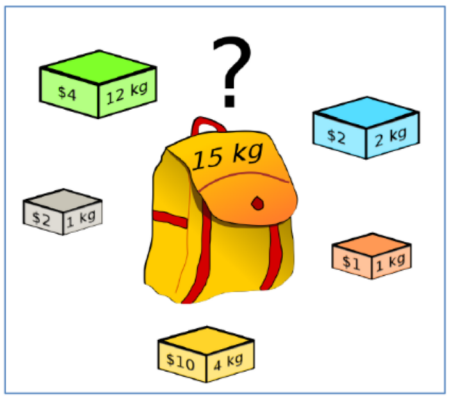
Formulating CSPs: Flight Gate Scheduling

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

7 flights

1. VARIABLES → $F_1, F_2, \ldots, F_7$
2. DOMAINS → Gate Nos $\{G_1, G_2, G_3, G_4\}$
3. SATISFACTION CONSTRAINTS → #flights having overlapping times cannot be assigned the same gate
4. OPTIMIZATION CRITERIA → Number of gates
5. SOLUTION

Formulating CSPs: Knapsack



$S = \{s_1, s_2, \ldots, s_n\}$

$W = \{w_1, w_2, \ldots, w_n\}$

$V = \{v_1, v_2, \ldots, v_n\}$

C : capacity

1. VARIABLES
2. DOMAINS
3. SATISFACTION CONSTRAINTS
4. OPTIMIZATION CRITERIA
5. SOLUTION

Variables: $s_1, s_2, \ldots, s_n$

Domain := $\{0, 1\}$ ✓

$\sum_{i=1}^{n} (s_i \cdot w_i) \leq C$ ✓

$\max \left(\sum_{i=1}^{n} s_i \cdot v_i \right)$

Formulating CSPs: Time Table

1. VARIABLES
2. DOMAINS
3. SATISFACTION CONSTRAINTS
4. OPTIMIZATION CRITERIA
5. SOLUTION

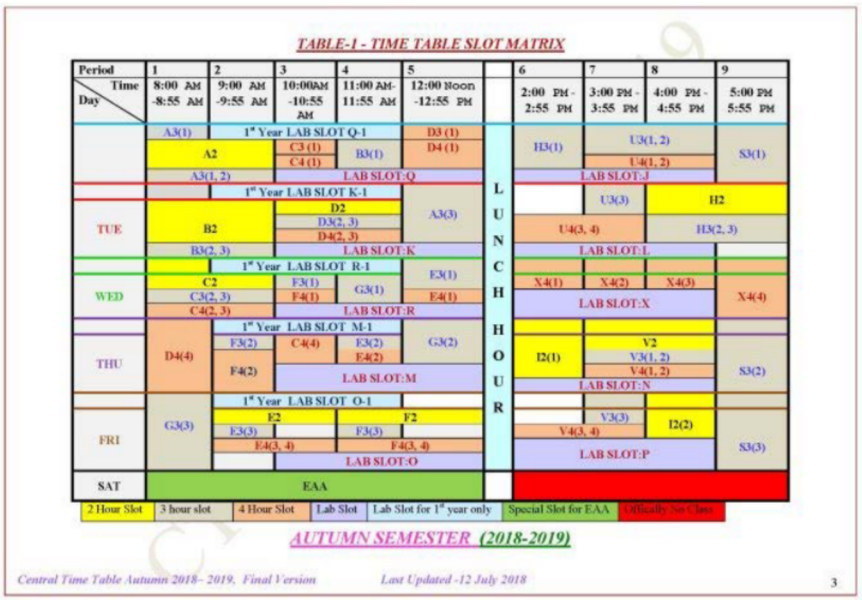


TABLE-1 - TIME TABLE SLOT MATRIX

Day \ Period / Time	1 8:00 AM -8:55 AM	2 9:00 AM -9:55 AM	3 10:00AM -10:55 AM	4 11:00 AM-11:55 AM	5 12:00 Noon -12:55 PM		6 2:00 PM - 2:55 PM	7 3:00 PM - 3:55 PM	8 4:00 PM - 4:55 PM	9 5:00 PM 5:55 PM
	A3(1)	1st Year LAB SLOT Q-1			D3 (1)		H3(1)	U3(1, 2)		S3(1)
	A2		C3 (1) C4 (1)	B3(1)	D4 (1)			U4(1, 2)		
	A3(1, 2)		LAB SLOT:Q				LAB SLOT:J			
		1st Year LAB SLOT K-1			A3(3)			U3(3)	H2	
TUE	B2		D2 D3(2, 3)			L U N C H H O U R	U4(3, 4)		H3(2, 3)	
	B3(2, 3)		LAB SLOT:K				LAB SLOT:L			
		1st Year LAB SLOT R-1			E3(1)					
WED	C2		F3(1)	G3(1)	E4(1)		X4(1)	X4(2)	X4(3)	X4(4)
	C3(2, 3)		F4(1)				LAB SLOT:X			
	C4(2, 3)		LAB SLOT:R							
		1st Year LAB SLOT M-1			G3(2)					
THU	D4(4)	F3(2)	C4(4)	E3(2)			I2(1)	V2 V3(1, 2)		S3(2)
		F4(2)	LAB SLOT:M	E4(2)				V4(1, 2)		
		1st Year LAB SLOT O-1								
FRI	G3(3)	E2		F2				V3(3)	I2(2)	S3(3)
		E3(3)		F3(3)			V4(3, 4)			
		E4(3, 4)		F4(3, 4)						
		LAB SLOT:O					LAB SLOT:P			
SAT	EAA									

2 Hour Slot | 3 hour slot | 4 Hour Slot | Lab Slot | Lab Slot for 1st year only | Special Slot for EAA | Officially Not Class

AUTUMN SEMESTER (2018-2019)

Central Time Table Autumn 2018 - 2019, Final Version Last Updated -12 July 2018 3

Slots, Rooms, Subjects, Teachers, Students

Room-Slots: Subjects
Subjects: L-T-P, Teachers, Students

Multi-layered constraints ✓

Intricate Optimization — free slots, total duration, minimize movement time

Exercise: Time-Tabling in the era of online classes

3 lectures/week

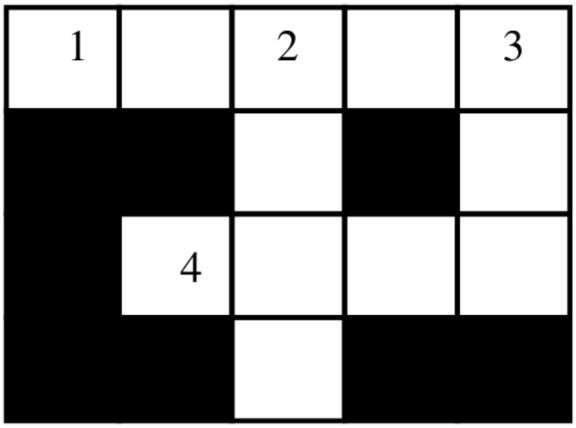
1-1-1 1-2

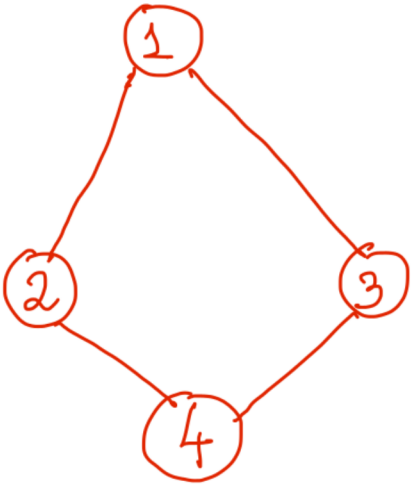
CSP Solution Overview

- **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

logic problem : satisfiability

CSP Graph for Crossword



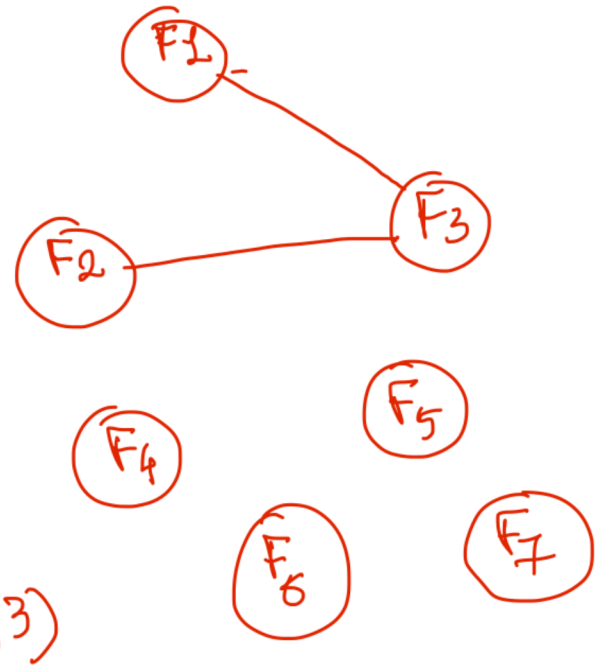


Word List:

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

CSP Graph 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



3 Gates (1, 2, 3)

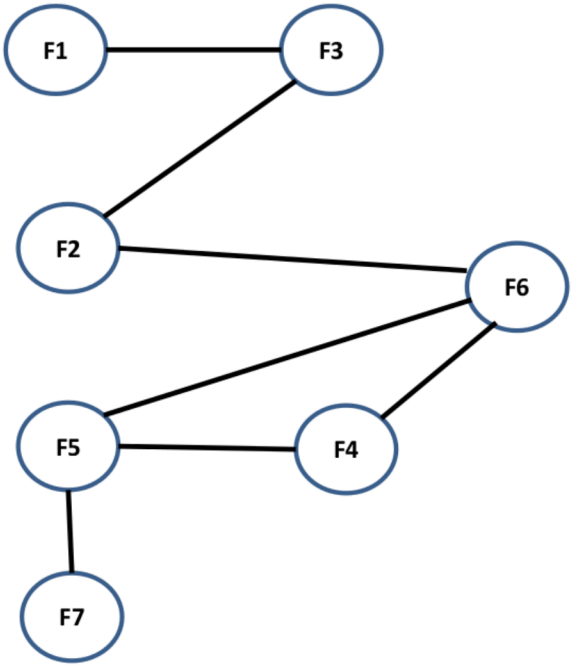
CSP Graph for Airline Gate Scheduling

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

{1, 2, 3}

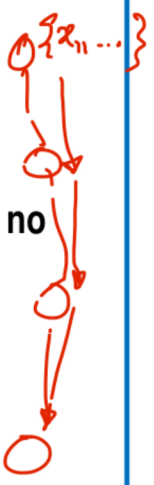
minimize

Dom: 7 Gates

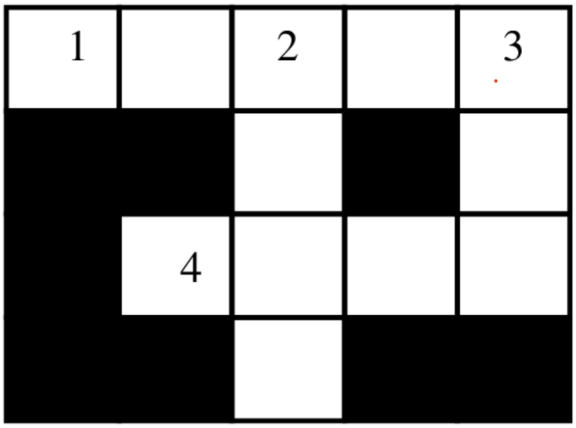


Constraint Propagation Steps

- **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** Edge 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
1.   queue ← local variable initialized to all arcs in csp
2.   while queue is not empty do
3.      (X_i, X_j) ← pop(queue)
4.      if Revise(csp, X_i, X_j) then
5.         if size of D_i = 0 then return false
6.         for each X_k in X_i.neighbors-{X_j} do
7.            add (X_k, X_i) to queue
8.   return true
```

Q = { all edges }

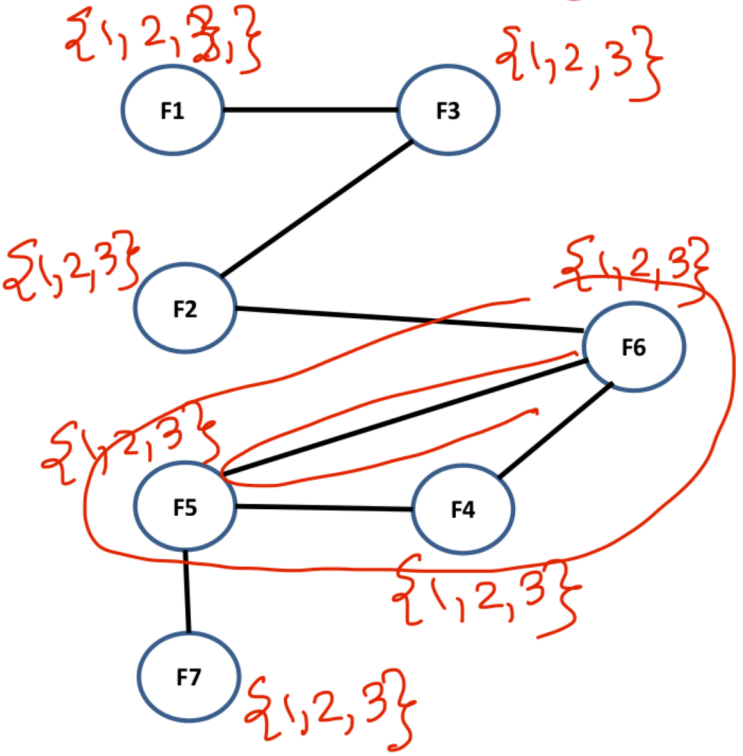
```
Revise(csp, X_i, X_j)
1.   revised ← false
2.   for each x in D_i do
3.      if no value y in D_j allows (x, y) to satisfy constraint between X_i and X_j then
4.         delete x from D_i
5.         revised ← true
6.   return revised
```

Time complexity: $O(n^2d^3)$

Consistency 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

{1,2} Arc



Backtracking Algorithm for CSP

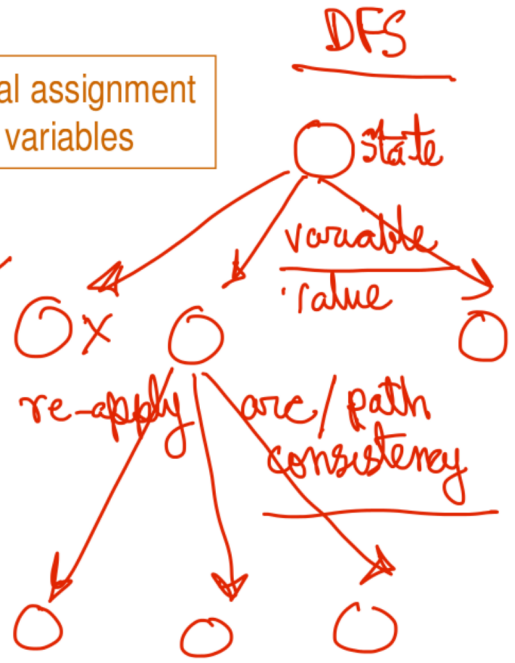
search

CSP-BACKTRACKING({})

CSP-BACKTRACKING(a)

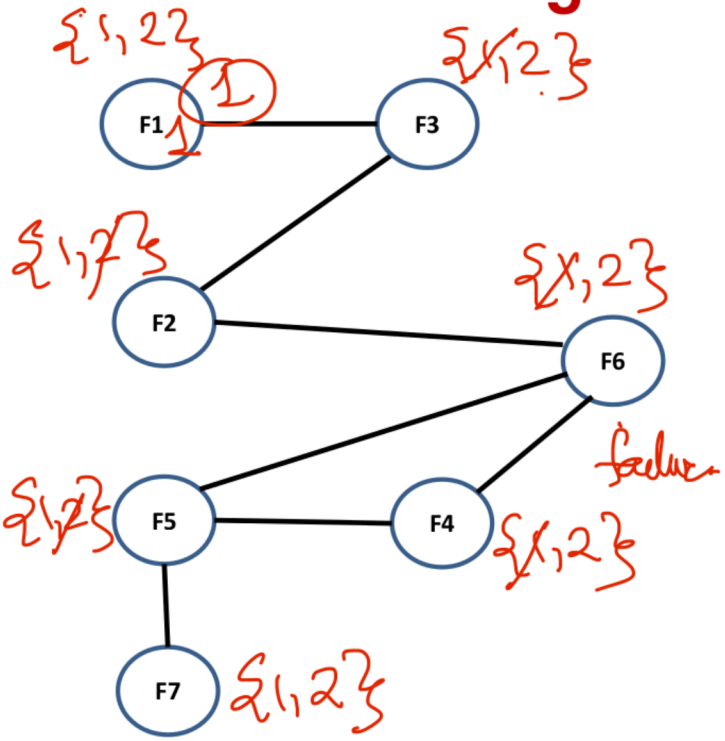
- If a is complete then return a
- X ← select unassigned variable
- D ← select an ordering for the domain of X
- For each value v in D do
  - If v is consistent with a then
    - Add (X= v) to a
    - result ← CSP-BACKTRACKING(a)
    - If result ≠ *failure* then return result
- Return *failure*

partial assignment of variables



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

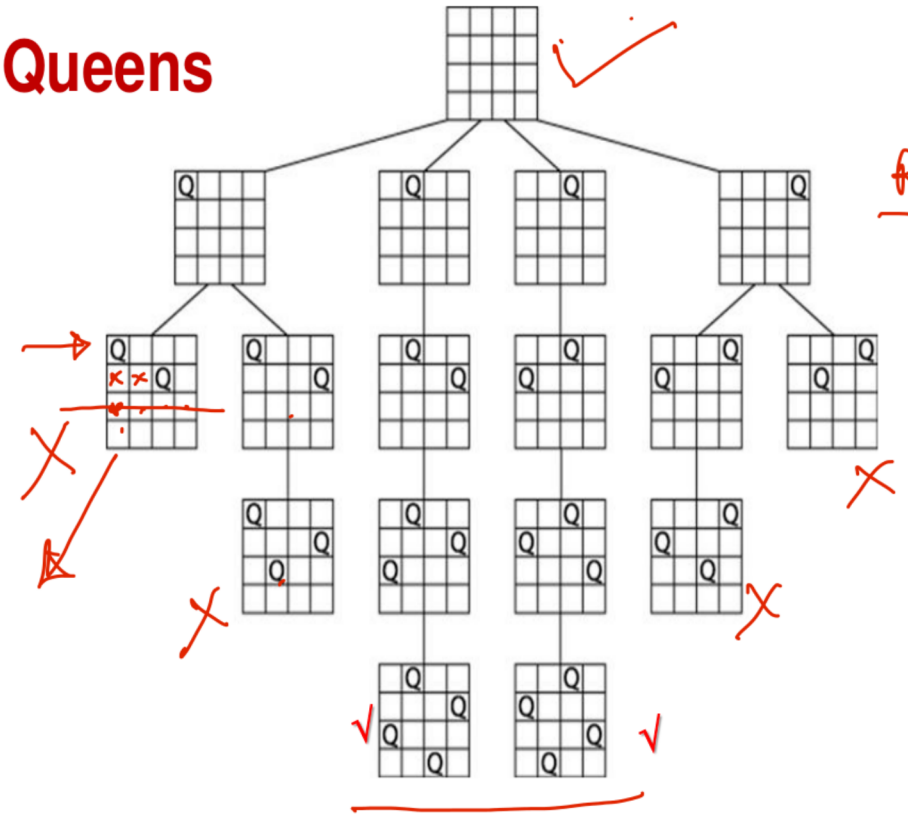


Search 4-Queens

DFS

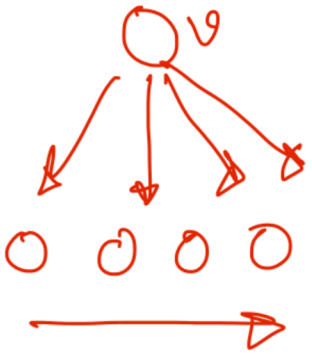
Row 1

Row 2



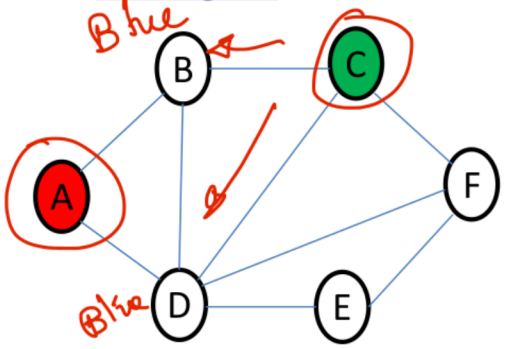
Strategies for CSP Search Algorithms

- 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



Forward Checking: 3 Colouring Problem

- Forward checking propagates information from assigned to unassigned variables



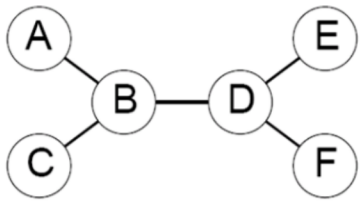
A	B	C	D	E	F
{R, G, B}	{R, G, B}	{R, G, B}	{R, G, B}	{R, G, B}	{R, G, B}
	{G, B}	{R, G, B}	{G, B}	{R, G, B}	{R, G, B}
	{B}		{B}	{R, G, B}	{R, B}

X

- B and D cannot both be blue!
- Why did we not detect this?
- Forward checking detects some inconsistencies, not all
- *Constraint propagation:* reason from constraint to constraint

Special Cases

Tree-structured CSPs



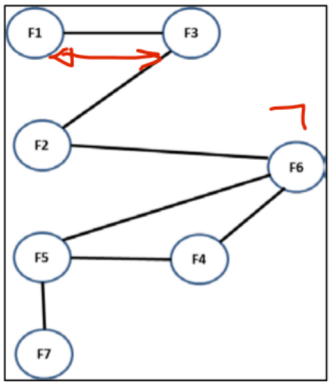
Theorem: if the constraint graph has no loops, the CSP can be solved in $O(n\,d^2)$ time

Compare to general CSPs, where worst-case time is $O(d^n)$

For PERFECT GRAPHS, CHORDAL GRAPHS, INTERVAL GRAPHS, the Graph Colouring Problem can be solved in Polynomial Time

Solving CSP using SAT / SMT Solvers

- Boolean Satisfiability (SAT) is a CSP
- CSPs can be modelled as SAT problems
  - Try: Map Colour, Gate Scheduling, n-Queens
  - Home Exercise: Write a Generic Scheme to Convert and CSP Problem to a SAT Problem
- SAT has very efficient solvers
  - MiniSAT, CHAFF, GRASP, etc
- For Optimization cases, we can formulate them as
  - Satisfiability Modulo Theories (SMT) – with arithmetic and first order logic
  - 0/1 or Integer Linear Programming (ILP)
  - Constraint Programming Problems
  - SMT Solvers: Z3, Yices, Barcelogic, MathSAT, OpenSMT, etc



Satisfying solution

$(f_{11} \wedge f_{21}) \wedge$

$\neg (f_{12} \wedge f_{22}) \wedge$

$\neg (f_{13} \wedge f_{23})$

Boolean Variables

F1: f_{11} f_{12} f_{13}

F2: f_{21} f_{22} f_{23}

$(f_{11} \vee f_{12} \vee f_{13}) \wedge (\quad)$

Thank you