Constraint Satisfaction Problem

- Till now we did Atomic search, now we look inside State.
- ~~A~~ State is no longer Atomic
- Constraint ~~satis~~ satisfaction problem has 3 components
  a) X Set of variable
  b) D set of Domains
  c) C set of Allowed combination of Values

Each constraint consists of <scope, rel> pair, scope is tuple of variable that participate in constraint and rel is relation that defines values of these variables

→ CSP deals with assignment of variables
  – No constraint violated : Complete Assignment
    ↳ Such type are ~~Comple~~ consistent / legal assignment

→ Partial assignment : Some constraint failed.

CSP is NP ~~bee~~ hard problems

Why CSP?

1. ~~Pur~~ Prune state space
2. Easy to formulate
3. Fast & efficient CSP solver present

→ In atomic search, we were looking for Goal state in here, if we find a partial state, we can discard it and its child as we know if any constraint violated solution never exists there.

Many states are untrackable for atomic state space search but can be solved quickly when formulated as CSP

- CSP used in Job scheduling

Types of Constraints

- Precedence constraints : before T_2 , T_1 must finish
- Disjunctive Constraints : If they have common tool must not overlap in time

- Simplest CSP are Discrete & Finite
  Ex Map Coloring problems

→ Discrete Domain Can be infinite like set of integers ~~on st~~ } use implicit constraint

→ linear Constraint Each variable appears in linear form (solvable)

→ Non linear - Undecidable problem

→ Constraint Satisfaction on Continuous Domains
  Ex Hubble space Telescope

Continuous domain CSPs Example : linear programming
[Constraint must be linear equality / inequality]

Can be solved in polynomial time.

→ Constraint based on type of Variable
- Unary Constraint : Restrict value of single variable
  Ex SA ≠ Green
  ↳ ⟨(SA), SA ≠ green⟩

→ Binary Constraint : One with unary and binary constraints
Can be represented with Constraint graph.
Ex SA ≠ NSW

→ Higher order ~~graph~~ constraint :
Ex $X < Y < Z$ or $X > Y > Z$

***

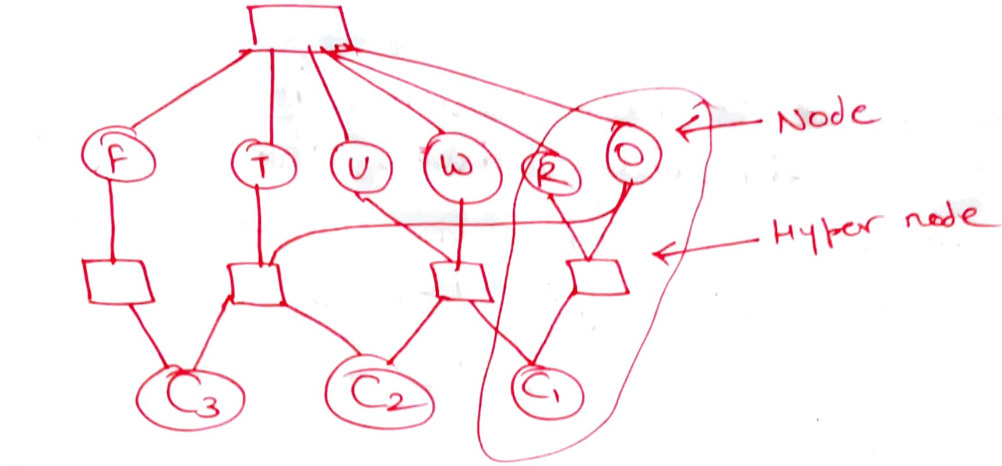
Global Constraint : It is NOT Constraint over All the variables
its Constraint on Arbitrary no. of variables

Ex Alldiff Constraint in Sudoku

OR

Cryptoarithmatic puzzle : Each letter represent diff digits Ex

TWO
TWO
FOUR } (Rules)



Constraint hyper graph

Constraints

- $O + O = R + 10 \cdot C_1$
- $C_1 + W + W = U + 100\ C_2$
- $C_2 + T + T = O + 1000\ C_3$
- $C_3 = F$

Reasons why we prefer Global Constraint over binary?

1. Easier and less Error prone
2. It is possible to design special purpose inference Algorithm for global constraints

So far we talked about Absolute Constraints, violation of which Rules out potential solution

Another type is Prefrence Constraints means if solution follow's this then its preffered
we call such problems Constrained optimization problem

- Constraint propogation inferences in CSP's

- CSP can do specific type of inference called constraint propogation using Constraint to reduce no. of legal moves
- It will leave fewer choices to consider when we make next choice
- Constraint propogation can be done
  - within Search
  - preprocessing [sometimes its enough to solve whole problem]

• Node consistency

→ Single variable is Node Consistent if all values in Variables Domain satisfy variables unary constraint

Ex S.A dislike Green, so its Domainset is changed from {G, B, R} $\xrightarrow{(to)}$ {B, R}

• A graph is node consistent if every variable is node consistant

→ ARC-Consistent

X_i is arc consistant on X_j iff every value of X_i in Domain D_i ~~is~~ has corresponding value D_j in X_j, that satisfy binary constraint on arc (X_i, X_j)

Ex $y = x^2$ < (X, Y), {(0,0), (1,1), (2,4), (3,9)} >

X's Domain (Arc Consistent w.r.t Y) = {0, 1, 2, 3}

Y's " (" " " X) = {0, 1, 4, 9}

For Australia SA, WA Example, Arc consistency has no effect (book page 170)

Most popular Arc consistency Algo - AC-3

[maintains Queue of Arc, initially Queue is ~~empty~~ has all Arcs in CSP, it pops them out one by one, makes ~~if~~ X_i is Arc consistent w.r.t X_j, If D_i is unchanged, Algo moves to next Arc]

[If it makes reduction, we readd this X_k for All X_i in Queue] (might change D_i / D_k again)

Repeat till no more Arc's in Queue

(Solution/search space gets smaller Drastically)

[in Some cases, it will completely solve ~~by~~ reducing size to ~~zero~~ one and in other it prove no solution by putting size zero]

AC-3

CSP of n variables ~~running~~ Domain size d times (at most) with c binary arcs, each arc can be inserted to Queue d times as X_i has atmost d value, so checking Consistency can be done in $O(d^2)$

worst case time: $O(cd^3)$

Path Consistency

→ with 2 colours Arc Consistency can be satisfied for australia but no solution exist (need atleast -3)

Map colouring needs stronger notion, Path consistency this tightens binary constraints (Arc) with implicit Constraint inferred by looking at triples of variables

[two variable set (X_m X_j) is path consistent with respect to 3rd variable X_m for every assignment of X_i X_j there exists valid assignment of X_m Such that $\{X_i\ X_m\}$ and $\{X_m\ X_j\}$ are satisfied]

K-Consistency (NP-complete)

A CSP is k consistent if it is also (k-1) (k-2)..... 1 Consistent [make 1 then 2 then 3.... n consistent]

Run time → $O(n^2d)$ worst case: (Exponential in n)

Constraint Satisfaction is NP complete

⇒ 2 Consistency & 3 Consistency are Common and less common respectively

Global Constraint

(Aribitrary no of constraint and not necessarily all)

one simple inconsistency detection: if m variable are involved and there are n possible distinct values altogther and m > n then constraint can not be Satisfied

(Read Algo page 178)

- Simple consistency sometimes is more effective over Arc consistency over binary constraints on Higher order Constraints

→ Resource constraint : Detect inconsistency by simply checking Sum of Min values of Current Domain
Also called Atmost constraint.
[Delete max value if not consistent with min of other Domain]

Large resource limited problems, its not possible to respresent large set so we use bounds (upper & lower) insted and call it bound propogation. [widely used in practical constraint problems]

~~Sudoku - 81 puzzle variables~~
~~27 Alldiff constraints~~

Backtracking for CSPs

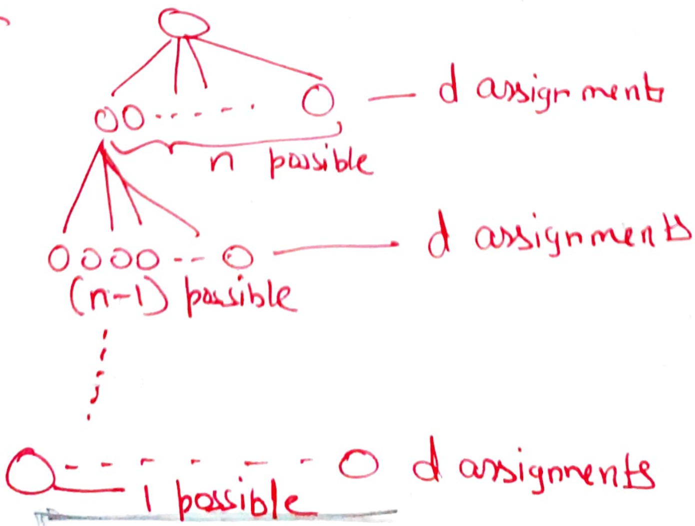
(2) Partial assignment — Backtracking
Complete assignment — local search

(1) If after constraint propogation, we still have multiple values then we use search

as discussed in class ⇝ $N! d^n$ leaves but only d^n complete assignments possible

$n! d^n$

why?
↳ CSPs are Commutative so actually d^n Unique only exists



Simplest Strategy – static ordering ⇝ randomly chose

intuitive way ⇝ $\underbrace{\text{Variable with least possible options}}$

M.R.V

(minimum Remeining value) or Most constrained or fail first

Chose in terms of value choice a variable has { Select it cause it will most likely cause failure otherwise
So select it and prune failure options

tie breaker
(degree heuristic) { M.R.V does not help selecting first value
So select the variable that constraints other unassigned variables the most

Least Constraining value (fail last)
↳ Pefer value that Rules out fewest choices for neighbouring values/variables
(trying to leave Max flexibility)

So
Variable we check fail first
Value we check fail last
→ Since we need one Solution, value ordering is irrelevent, to find All Solutions its relevent

- Inference is More powerful than AC-3
Simplest inference – forward Checking
Assign a value, then reduce/update everyone elses domains

Most problems ⟹ MRV + forward Chk is effective

forward checking does not detect All inconsistencies
(Example in book 178)

• Maintaining Arc consistency Algo detect inconsistency

when in AC-3 any set becomes empty, we backtrack as empty set over unassigned Variable is inconsistency

→ Backtracking is a possible way but its silly, many similar will come (Discussed before)

Solution
→ keep track of Conflicting set
→ forward checking is better than back tracking redundent can be prooned

A backtracking Algo that uses conflict set is called Conflict - directed backtracking

Constraint learning
↳ Find set of problems that are sure to cause ~~cont~~ conflict / failure

min set of variables from conflict set called "no-good"

Local search in CSP's
used in n-Queens
We use value that result in min no. of conflicts
[How tabu search came in place]

Constraint weighting
increase +ve ones and decrease -ve once

Constraint graph $\xrightarrow{\text{break}}$ independent sub graphs

by finding Connected components

$\Downarrow$ $\qquad\qquad$ $\Downarrow$

$O(d^n)$ time $\xrightarrow[\text{breaking}]{\text{after}}$ $O(d^c\, n/c)$ time

Tree structure can be solved in linear no. of variables

↳ notion called directional Arc consistency

↳ Done in topological sort ordering

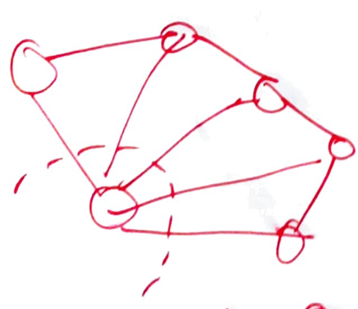
time → $O(nd^2)$ (page 183)

Convert CSP to tree CSP

Cutset

(pick a subset of variables, remove them and their edges, by assigning them and reducing others domain

So rest is tree easy to solve



tree decomposition

→ Every variable appear atleast once

→ two variables constrained should come togther in atleast one node

→ If a variable appears in tree at two nodes, it must appear in every node along path connecting node

(186)

[Break into small trees]

After decomposition we apply Algo, it takes $O(nd^2)$

no. of nodes (n), largest domain (d)

$T.C = O(d^c (n-c)\, d^2)$

Smallest cycle in Cuset is NP ~~hard~~ hard

(Can be reduced by cuset Conditioning)

If a graph ~~as~~ has width ω, it can be solved in $O(nd^{\omega+1})$ time

finding decomposition is NP hard

Q- whats better cutset or decomposition

time wise - tree better (cutset has exponential)

memory wise - cutset better (tree has exponential)