Elective Course
Course Code: CS4103
Autumn 2025-26

Lecture #18

Artificial Intelligence for Data Science

Week-5: CONSTRAINT SATISFACTION PROBLEM (CSP) [Part-IV]
(Solving CSPs)

Course Instructor:
Dr. Monidipa Das
Assistant Professor
Department of Computational and Data Sciences
Indian Institute of Science Education and Research Kolkata, India 741246

Backtracking Search for Solving CSP

2

function BACKTRACKING-SEARCH(*csp*) **returns** a solution, or failure
  return BACKTRACK({ }, *csp*)

function BACKTRACK(*assignment*, *csp*) **returns** a solution, or failure
  if *assignment* is complete **then return** *assignment*
  var ← SELECT-UNASSIGNED-VARIABLE(*csp*)
  for each *value* **in** ORDER-DOMAIN-VALUES(*var*, *assignment*, *csp*) **do**
    if *value* is consistent with *assignment* **then**
      add {*var* = *value*} to *assignment*
      inferences ← INFERENCE(*csp*, *var*, *value*)
      if *inferences* ≠ *failure* **then**
        add *inferences* to *assignment*
        result ← BACKTRACK(*assignment*, *csp*)
        if *result* ≠ *failure* **then**
          return *result*
    remove {*var* = *value*} and *inferences* from *assignment*
  return *failure*

Comparison of CSP Algorithms

3

Comparison on the consistency check counted over five runs

Problem	Backtracking	BT+MRV	Forward Checking	FC+MRV
USA	(> 1,000K)	(> 1,000K)	2K	60
n-Queens	(> 40,000K)	13,500K	(> 40,000K)	817K
Zebra	3,859K	1K	35K	0.5K

Street Puzzle

4

1 2 3 4 5

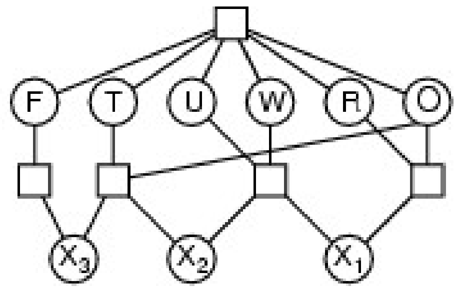
N_i = {English, Spaniard, Japanese, Italian, Norwegian}
C_i = {Red, Green, White, Yellow, Blue}
D_i = {Tea, Coffee, Milk, Fruit-juice, Water}
J_i = {Painter, Sculptor, Diplomat, Violinist, Doctor}
A_i = {Dog, Snails, Fox, Horse, Zebra}

The Englishman lives in the Red house
The Spaniard has a Dog
The Japanese is a Painter
The Italian drinks Tea
The Norwegian lives in the first house on the left
The owner of the Green house drinks Coffee
The Green house is on the right of the White house
The Sculptor breeds Snails
The Diplomat lives in the Yellow house
The owner of the middle house drinks Milk
The Norwegian lives next door to the Blue house
The Violinist drinks Fruit juice
The Fox is in the house next to the Doctor's
The Horse is next to the Diplomat's

Who owns the Zebra?
Who drinks Water?

Cryptarithmetic Puzzle

```
  T W O
+ T W O
-------
F O U R
```

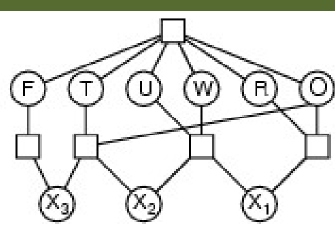


- **Variables:** *F T U W R O* $X_1\ X_2\ X_3$
- **Domains:** {0,1,2,3,4,5,6,7,8,9}
- **Constraints:** *Alldiff (F,T,U,W,R,O)*

$O + O = R + 10 \cdot X_1$
$X_1 + W + W = U + 10 \cdot X_2$
$X_2 + T + T = O + 10 \cdot X_3$
$X_3 = F,\ T \neq 0,\ F \neq 0$

Solution

```
  T W O
+ T W O
-------
F O U R
```



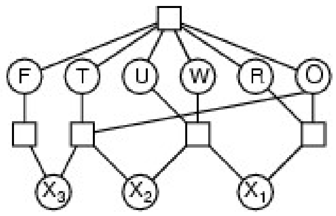
$O + O = R + 10 \cdot X_1$
$X_1 + W + W = U + 10 \cdot X_2$
$X_2 + T + T = O + 10 \cdot X_3$
$X_3 = F,\ T \neq 0,\ F \neq 0$
Alldiff (F,T,U,W,R,O)

X1	X2	X3	F	T	U	W	R	O
{0,1}	{0,1}	{0,1}	{1,2,3,4,5,6,7,8,9}	{1,2,3,4,5,6,7,8,9}	{0,1,2,3,4,5,6,7,8,9}	{0,1,2,3,4,5,6,7,8,9}	{0,1,2,3,4,5,6,7,8,9}	{0,1,2,3,4,5,6,7,8,9}
{0,1}	{0,1}	{1}	{1}	{1,2,3,4,5,6,7,8,9}	{0,1,2,3,4,5,6,7,8,9}	{0,1,2,3,4,5,6,7,8,9}	{0, 2, 4, 6, 8}	{0,1,2,3,4,5,6,7,8,9}
{0,1}	{0,1}	1	1	{2,3,4,5,6,7,8,9}	{0,2,3,4,5,6,7,8,9}	{0,2,3,4,5,6,7,8,9}	{0, 2, 4, 6, 8}	{0,2,3,4,5,6,7,8,9}
{0,1}	0	1	1	{2,3,4,5,6,7,8,9}	{0,2,3,4,5,6,7,8,9}	{0,2,3,4,5,6,7,8,9}	{0, 2, 4, 6, 8}	{0, 2, 4, 6, 8}
0	0	1	1	{5, 6, 7}	{0, 4, 6, 8}	{0, 2, 3, 4}	{0, 4, 8}	{0, 2, 4}
0	0	1	1	{6, 7}	{4, 6, 8}	{2, 3, 4}	{4, 8}	{2, 4}
0	0	1	1	7	{6}	{2, 3}	{8}	{4}
0	0	1	1	7	{6}	{3}	{8}	{4}
0	0	1	1	7	6	3	8	4

Solution

7

```
  T W O
+ T W O
F O U R
```



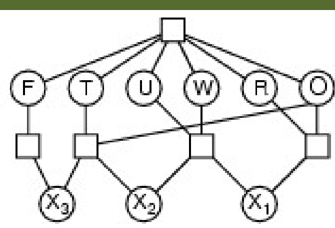
$O + O = R + 10 \cdot X_1$
$X_1 + W + W = U + 10 \cdot X_2$
$X_2 + T + T = O + 10 \cdot X_3$
$X_3 = F, T \neq 0, F \neq 0$
Alldiff (F,T,U,W,R,O)

X1	X2	X3	F	T	U	W	R	O
{0,1}	{0,1}	{0,1}	{1,2,3,4,5,6,7,8,9}	{1,2,3,4,5,6,7,8,9}	{0,1,2,3,4,5,6,7,8,9}	{0,1,2,3,4,5,6,7,8,9}	{0,1,2,3,4,5,6,7,8,9}	{0,1,2,3,4,5,6,7,8,9}
{0,1}	{0,1}	{1}	{1}	{1,2,3,4,5,6,7,8,9}	{0,1,2,3,4,5,6,7,8,9}	{0,1,2,3,4,5,6,7,8,9}	{0, 2, 4, 6, 8}	{0,1,2,3,4,5,6,7,8,9}
{0,1}	{0,1}	1	1	{2,3,4,5,6,7,8,9}	{0,2,3,4,5,6,7,8,9}	{0,2,3,4,5,6,7,8,9}	{0, 2, 4, 6, 8}	{0,2,3,4,5,6,7,8,9}
{0,1}	0	1	1	{2,3,4,5,6,7,8,9}	{0,2,3,4,5,6,7,8,9}	{0,2,3,4,5,6,7,8,9}	{0, 2, 4, 6, 8}	{0, 2, 4, 6, 8}
0	0	1	1	{5, 6, 7}	{0, 4, 6, 8}	{0, 2, 3, 4}	{0, 4, 8}	{0, 2, 4}
0	0	1	1	{6, 7}	{4, 6, 8}	{2, 3, 4}	{4, 8}	{2, 4}
0	0	1	1	6	{8}	{3}	{4}	{2}
					Failure!			

Another Solution

8

```
  T W O
+ T W O
F O U R
```



$O + O = R + 10 \cdot X_1$
$X_1 + W + W = U + 10 \cdot X_2$
$X_2 + T + T = O + 10 \cdot X_3$
$X_3 = F, T \neq 0, F \neq 0$
Alldiff (F,T,U,W,R,O)

X1	X2	X3	F	T	U	W	R	O
{0,1}	{0,1}	{0,1}	{1,2,3,4,5,6,7,8,9}	{1,2,3,4,5,6,7,8,9}	{0,1,2,3,4,5,6,7,8,9}	{0,1,2,3,4,5,6,7,8,9}	{0,1,2,3,4,5,6,7,8,9}	{0,1,2,3,4,5,6,7,8,9}
{0,1}	{0,1}	{1}	{1}	{1,2,3,4,5,6,7,8,9}	{0,1,2,3,4,5,6,7,8,9}	{0,1,2,3,4,5,6,7,8,9}	{0, 2, 4, 6, 8}	{0,1,2,3,4,5,6,7,8,9}
{0,1}	{0,1}	1	1	{2,3,4,5,6,7,8,9}	{0,2,3,4,5,6,7,8,9}	{0,2,3,4,5,6,7,8,9}	{0, 2, 4, 6, 8}	{0,2,3,4,5,6,7,8,9}
{0,1}	1	1	1	{6,7,8,9}	{0,2,3,4,5,6,7,8,9}	{0,2,3,4,5,6,7,8,9}	{0, 4, 6, 8}	{3, 5, 7, 9}
0	1	1	1	{6}	{0, 4, 8}	{5, 7, 9}	{6}	{3}
1	1	1	1	{7,8}	{3,5,7}	{6,7,8}	{0, 4}	{5, 7}
1	1	1	1	{7,8}	{3,7}	{6,8}	0	{5}
1	1	1	1	{7}	{3}	{6}	0	5
1	1	1	1	7	3	6	0	5

Local Search Problem: Definition

9

- **Definition:** A local search problem consists of a:
  - **CSP:** a set of variables, domains for these variables, and constraints on their joint values. A node in the search space will be a **complete assignment** to all of the variables.
  - **Neighbor relation:** an edge in the search space will exist when the neighbor relation holds between a pair of nodes.
  - **Scoring function:** $h(n)$, judges cost of a node (want to minimize)
    - E.g. the number of constraints violated in node n.
    - E.g. the cost of a state in an optimization context.

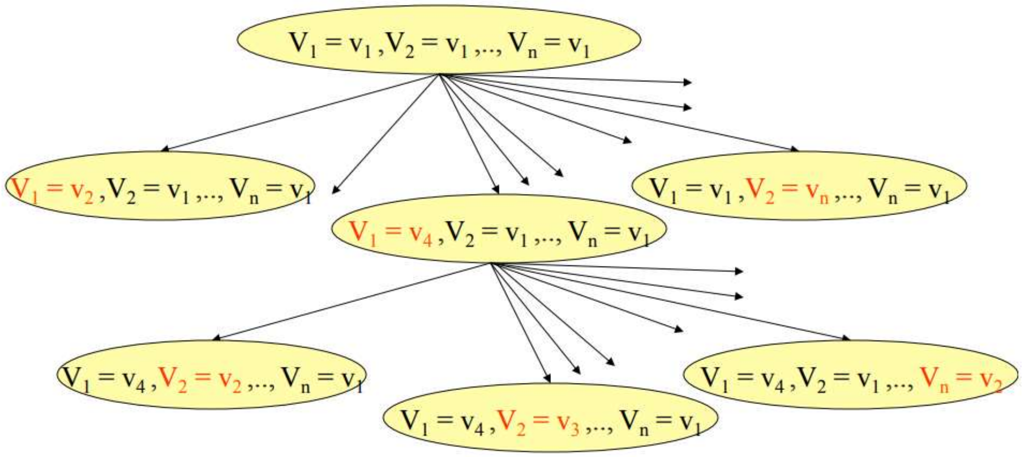
Local Search for CSPs

10

- Hill-climbing, simulated annealing typically work with "complete" states, i.e., all variables assigned
- To apply to CSPs:
  - allow states with unsatisfied constraints
  - operators reassign variable values
- Variable selection: randomly select any conflicted variable
- Value selection by min-conflicts heuristic:
  - choose value that violates the fewest constraints
  - i.e., hill-climb with $h(n)$ = total number of violated constraints

Search Space for Local Search

11

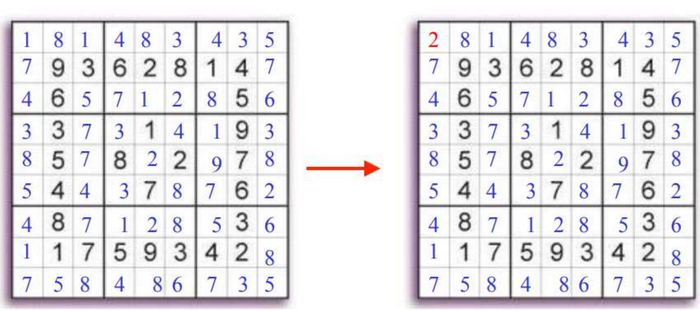


Local search does NOT backtrack!

Example: Sudoku as a local search problem

12

- CSP: usual Sudoku CSP
  - One variable per cell; domains {1,…,9};
  - Constraints: each number occurs once per row, per column, and per 3x3 box
- Neighbor relation: value of a single cell differs
- Scoring function: number of constraint violations



Local Search for CSP: MIN-CONFLICTS Algorithm

13

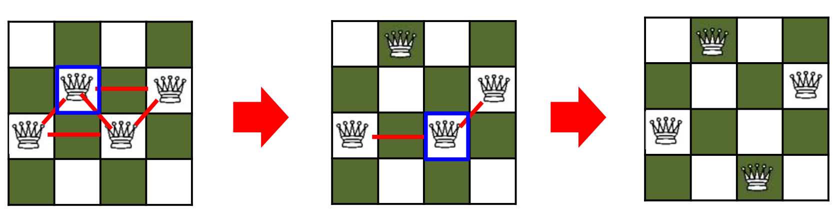
function MIN-CONFLICTS(*csp*, *max_steps*) **returns** a solution or failure
  inputs: *csp*, a constraint satisfaction problem
    max_steps, the number of steps allowed before giving up

  current ← an initial complete assignment for *csp*
  for *i* = 1 to *max_steps* **do**
    if *current* is a solution for *csp* **then return** *current*
    var ← a randomly chosen conflicted variable from *csp*.VARIABLES
    value ← the value *v* for *var* that minimizes CONFLICTS(*var*, *v*, *current*, *csp*)
    set *var* = *value* in *current*
  return *failure*

Example: 4-Queens

14

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



- Given random initial state, can solve n-queens in almost constant time for arbitrary n with high probability (e.g., n = 10,000,000)

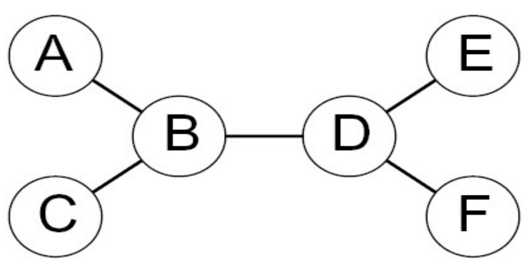
Solving CSPs: Improving Efficiency

15

- Which variable should be assigned next?
- In what order should its values be tried?
- Can we **detect** inevitable failures early?
- Can we take advantage of the **problem structure**?

Tree-structured CSPs

16



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

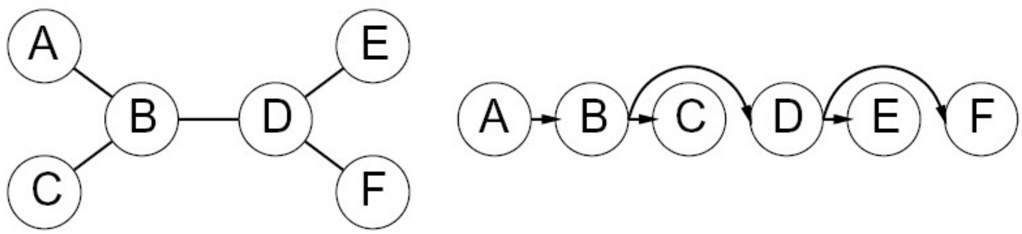
Compare to general CSPs, where worst-case time is $O(d^n)$

This property also applies to logical and probabilistic reasoning: an important example of the relation between syntactic restrictions and the complexity of reasoning.

Algorithm for tree-structured CSPs

17

1. Choose a variable as root, order variables from root to leaves such that every node's parent precedes it in the ordering



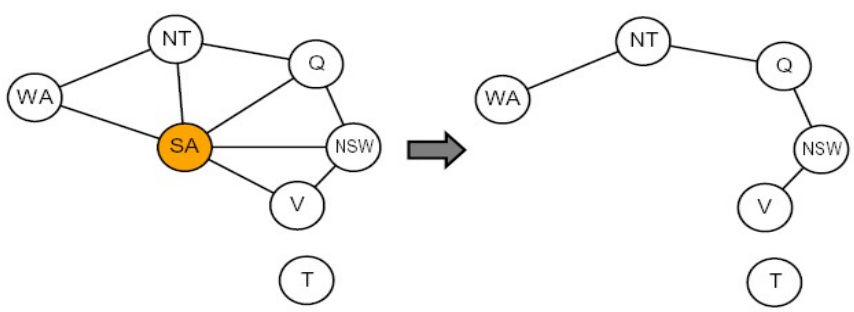
2. For j from n down to 2, apply $\text{RemoveInconsistent}(Parent(X_j), X_j)$

3. For j from 1 to n, assign X_j consistently with $Parent(X_j)$

Nearly tree-structured CSPs

18

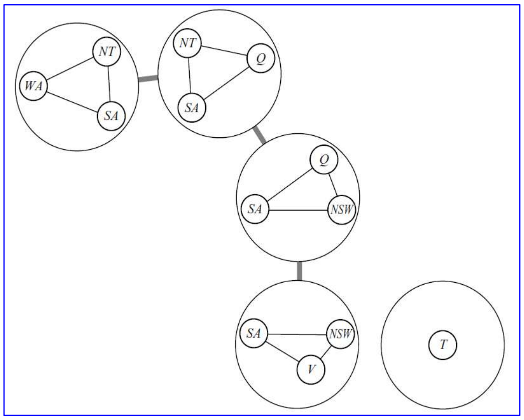
Conditioning: instantiate a variable, prune its neighbors' domains



Cutset conditioning: instantiate (in all ways) a set of variables such that the remaining constraint graph is a tree

Cutset size $c \Rightarrow$ runtime $O(d^c \cdot (n-c)d^2)$, very fast for small c

Tree decomposition



- Every variable in original problem must appear in at least one subproblem
- If two variables are connected in the original problem, they must appear together (along with the constraint) in at least one subproblem
- If a variable occurs in two subproblems in the tree, it must appear in every subproblem on the path that connects the two

- Algorithm: solve for all solutions of each subproblem. Then, use the tree-structured algorithm, treating the subproblem solutions as variables for those subproblems.
- $O(nd^{w+1})$ where w is the *treewidth* (= one less than size of largest subproblem)

Questions?