

AI: Sudoku problem using Constraint Satisfaction Search

Team Members: Dipneet Kaur (12) & Drishti Malhotra (13)

Components of Constraint Satisfaction Problem (CSP)

Components of Constraint Satisfaction Problem

- ② Variables
- ⑥ Domain
- ③ Constraints
- ④ Worlds
- ⑤ Models.

Using Sudoku and identifying these components:

Sudoku - Constraint Satisfaction problem

↑ has

Variables Domain Constraints

③ C₁ - Rows should not have duplicates
Sudoku Puzzle

③ C₂ - 3x3 grid should not have duplicates
Sudoku Solution

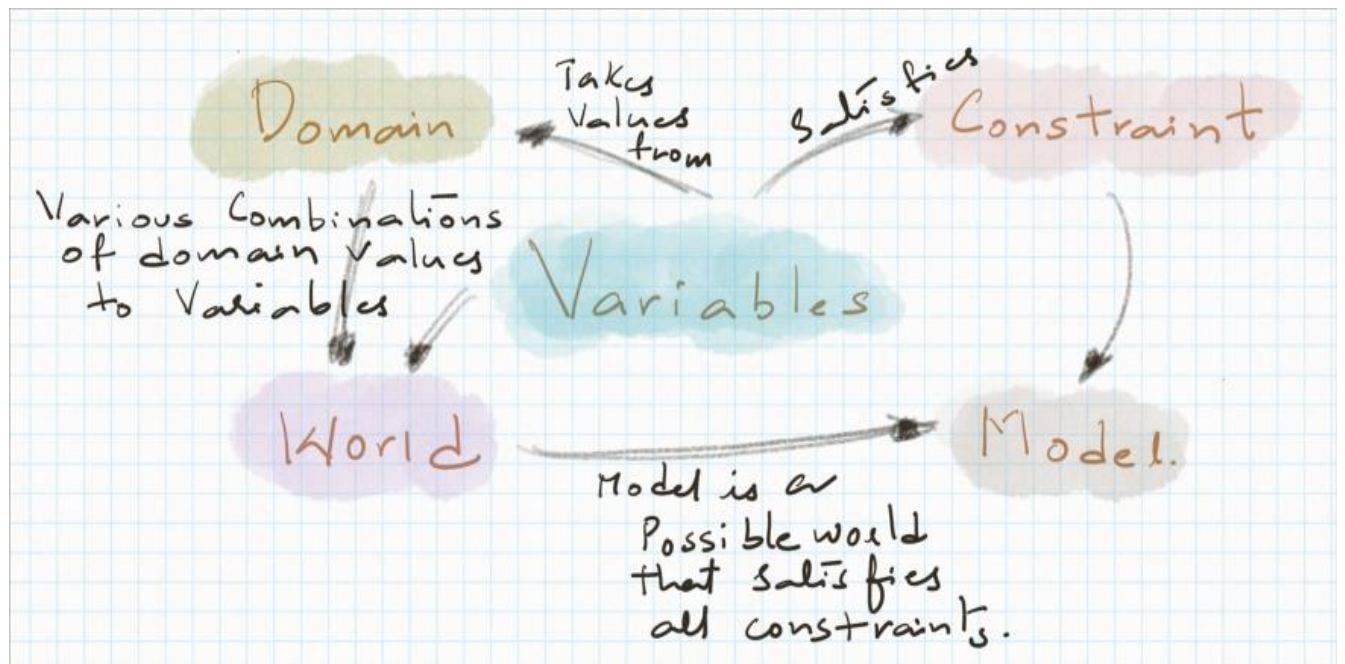
9	3	6	2	8	1	4	
6						5	
3			1			9	
5		8		2		7	
4			7			6	
8						3	
1	7	5	9	3	4	2	

③ C₃ - Column should not have duplicates

2	7	1	9	5	4	6	8	3
5	9	3	6	2	8	1	4	7
4	6	8	1	3	7	2	5	9
7	3	6	4	1	5	8	9	2
1	5	9	8	6	2	3	7	4
8	4	2	3	7	9	5	6	1
9	8	5	2	4	1	7	3	6
6	1	7	5	9	3	4	2	8
3	2	4	7	8	6	9	1	5

① Every empty cell is a variable

② Domain for the variable is {1, 2, 3, 4, 5, 6, 7, 8, 9}



Variables:

Rows are indexed alphabetically starting with {A, B, C, D .. and so on.}
 Columns are indexed by digits starting with {1,2,3,4 .. and so on.}
 For 9*9 sudoku boards, variables are {A1, A2...A9, B1, B2 ,..., I1,I2,...I9.}
 At max 81 variables can be present.

Domain:

For each variable, the value lies b/w [1,9]

Constraints:

Each row consists of 9 distinct values $\in [1,9]$.
 I.e. $\langle A1, A2..A9 \rangle$, (all should be distinct)> and so on.
 Each column consists of 9 distinct values $\in [1,9]$.
 I.e. $\langle A1, B1,...,I1 \rangle$, (all should be distinct)>
 For each 3*3 blocks (no overlapping blocks) starting from A1 cell, consists of the 9 distinct values $\in [1,9]$.
 I.e. $\langle A1, A2, A3, B1, B2, B3, C1, C2, C3 \rangle$, (all should be distinct)>

Brute force to get all possibilities:

Brute force → Generate all possible worlds and test if a world is a model.
Generate & test

Complexity $O(cd^n)$

① ② ② ← Variables
 ↑ ↑ ↑
 d values
 d^n - World C-lookups to check
 $O(cd^n)$ ← C constraints

1	2	3	4	5	6	7	8	9
10	9	3	6	2	8	1	4	11
12	6	13	14	15	16	17	5	18
19	3	20	21	1	22	23	9	24
25	5	26	8	27	2	28	7	29
30	4	31	32	7	33	34	6	35
36	8	37	38	39	40	41	3	42
43	1	7	5	9	3	4	2	44
45	4	46	47	48	49	50	5	51
52	53	54	55	56	57	58	59	60

Possible Worlds
 # Variables - 53
 # domains - 9
 # possible worlds 9^{53} .

Reducing the Search space:

CSP as SEARCH

Components of Search.

① States

(Partial) assignment
of values to
Variables

② Start state

Empty
Assignment

③ Goal state

Complete
Assignment.

④ Successor State

State with next Variable
assigned.

9	3	6	2	8	1	4
6						5
3		1				9
5	8		2			7
4		7				6
8						3
1	7	5	9	3	4	2

Start state
Initially when
puzzle is
given

Successor
state

2	9	3	6	2	8	1	4
6							5
3		1					9
5	8		2				7
4		7					6
8							3
1	7	5	9	3	4	2	

2	7	1	9	5	4	6	8	3
5	9	3	6	2	8	1	4	7
4	6	8	1	3	7	2	5	9
7	3	6	4	1	5	8	9	2
1	5	9	8	6	2	3	7	4
8	4	2	3	7	9	5	6	1
9	8	5	2	4	1	7	3	6
6	1	7	5	9	3	4	2	8
3	2	4	7	8	6	9	1	5

Goal state.

CSP Search using Backtracking:

Methodology:

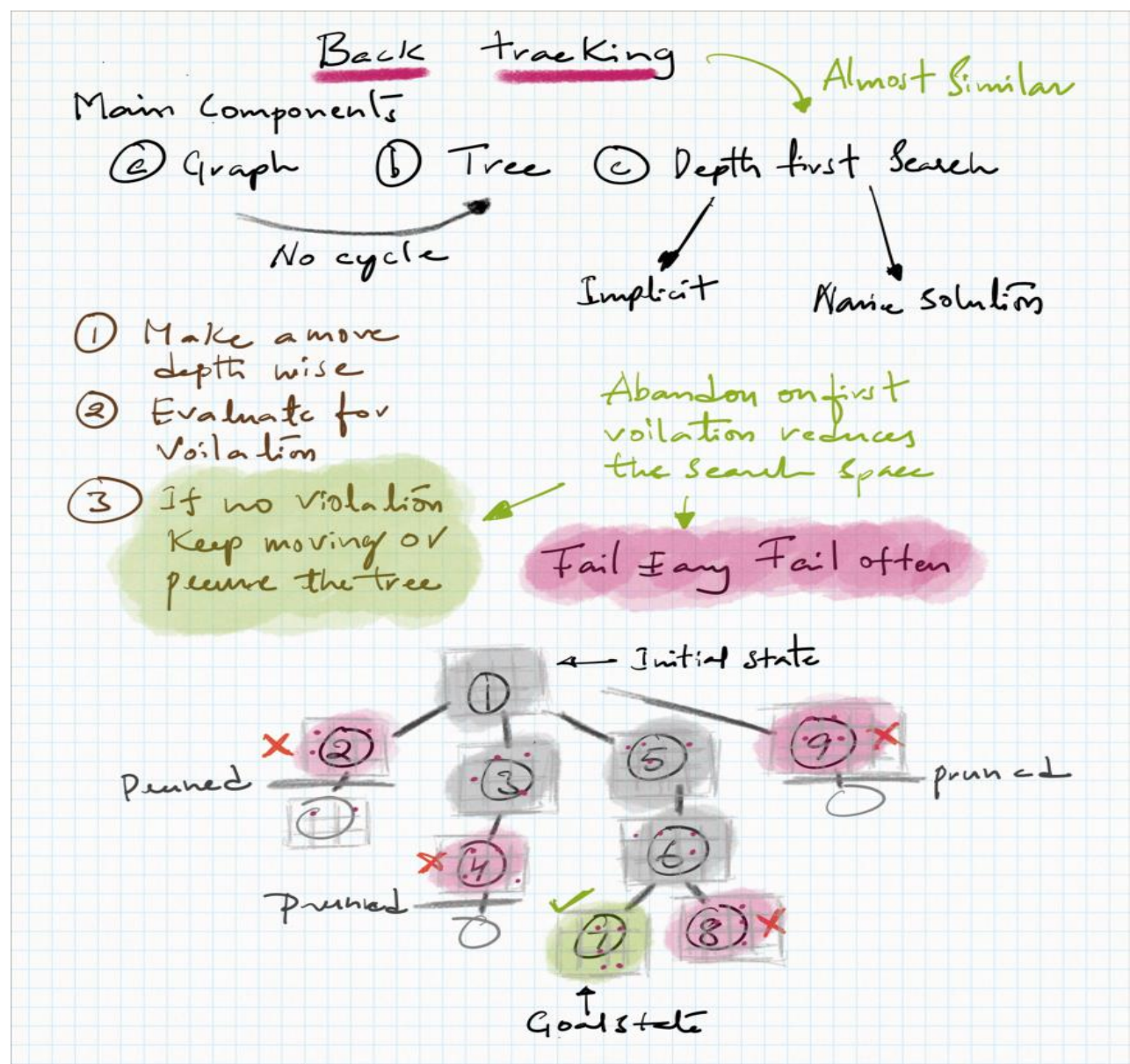
For each row, column and block of 3×3 We maintain a set which will tell us about the values which are already taken by the particular row, column and block.

When we assign a value to a cell, we find out a value which is not included in the underlying row, column and block, and assign that value to the cell.

In the same manner we try to assign values to all the cells until all cells are filled or a constraint violation has taken place.

In case of a constraint violation we backtrack and try to assign other values.

We keep on repeating these steps until all the cells are filled.



Using Constraint Propagation to solve CSP:

Constraint Propagation

	A	B	C	D	E	F	G	H	I
1	1,2 4,5	7	1,2 4,5	1,2,3 4,9				8	
2	5,7	9	3	6	2	8	1	4	7
3	1,2,4 8,5	6	1,2 4,8,5						
4		3		4	1	4,5		9	
5		5		8	3,4 6	2		7	
6		4		3,9	7	5,9		6	
7		8						3	
8		1	7	5	9	3	4	2	
9		2						1	

Variable A2
is dependent
on B2...I2,
A1...A9, B1, B3

That Means A2's
Constraint
involves variables
B2...I2, A1...A9,
B1, B3

If "7" is assigned to I2 then A2 is recomputed
A2 5,7 will have only "5" This recomputation
continues... till there are no changes it is
called constraint propagation.

One more example.

	D	E	F
4	4	1	4,5
5	8	3,4 6	2
6	3,9	7	5,9

Constraint
propagation

NO SEARCH

REDUCE DOMAIN
VALUES

	D	E	F
4	4	1	5
5	8	6	2
6	3	7	9

Generalization

For sudoku of sizes $16*16$, $25*25$, $36*36$, $n*n$ in general

We would need to assume n as a perfect square for making smaller blocks,
Let $m = \sqrt{n}$

Variables will increase depending on the board size (Number of variable = $n*n - \#(\text{given values})$)
At max $n*n$ variables can be present.

Domain for each cell will become $[1,n]$

Constraints:

For each row we need distinct values for each cell $\in [1,n]$

For each column we need distinct values for each cell $\in [1,n]$

For each block of $m*m$ we need distinct values for each cell $\in [1,n]$

Methodology will be the same as previous.

References:

www.medium.com

Thank You.