Constraint Satisfaction Problem

HU-HTAKM
Website: https://htakm.github.io/htakm_test/

November 5, 2024

1 Introduction to Constraint Satisfaction Problem

Definition 1.1. A **constraint satisfaction problem** is defined as:

1. $X_1, X_2, \cdots, X_n$: A set of variables
2. $D_1, D_2, \cdots, D_n$: A set of their corresponding domain of values ($X_i \in D_i$)
3. $C_1, C_2, \cdots, C_m$: A set of constraints

and the goal is to find a solution that satisfies all the constraints.

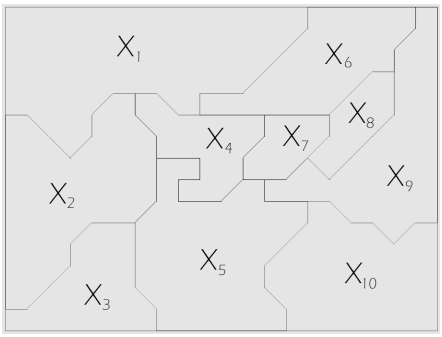
Example 1.1.1. We consider map colouring. Any two adjacent regions must not be painted with the same colour.

Variables:
$X_1, \cdots, X_{10}$

Domain:
$D_i = \{\text{Red}, \text{Green}, \text{Yellow}, \text{Brown}\}$ for $i = 1, \cdots, 10$

Constraint:
$X_1 \neq X_2$, $X_1 \neq X_4$, $X_1 \neq X_6$, $X_2 \neq X_3$, $X_2 \neq X_4$, $\cdots$

Goal:
Assign a colour to all variables that satisfy all constraints.



Example 1.1.2. We consider a simple cryptarithmetic puzzle. It should not have a leading zero.

Variables:
G, B, A, M, S, L, E

Domain:
$D_X = \{0, 1, \cdots, 9\}$ for $X = G, \cdots, E$, $D_{C_i} = \{0, 1\}$

Constraint:
Alldiff(G, B, A, M, S, L, E), $B \neq 0$, $G \neq 0$
$E + L = S + 10C_1$, $S + L + C_1 = E + 10C_2$
$2A + C_2 = M + 10C_3$, $2B + C_3 = A + 10C_4$, $G = C_4$,
where $C_1, C_2, C_3, C_4 \in \{0, 1\}$

$$\begin{array}{cccccc} & & B & A & S & E \\ + & & B & A & L & L \\ \hline & G & A & M & E & S \end{array}$$

Goal:
Assign a number to all variables that satisfy all constraints.

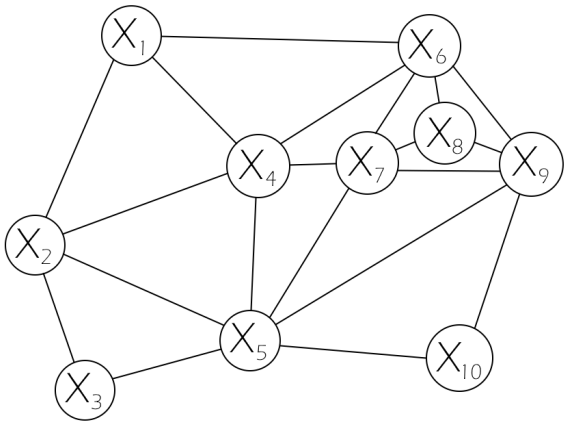
There are varieties of CSP Constraints. These involves:

1. **Unary constraint**: Involves only a single variable. E.g. $B \neq 0$
2. **Binary constraint**: Involves pairs of variables. E.g. $X_1 \neq X_2$
3. **High-order constraints**: Involves 3 or more variables. E.g. $E + L = S + 10C_1$

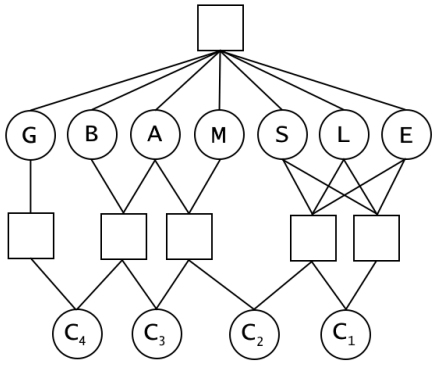
We can use a constraint graph to represent all the binary constraints and high-order constraints.

1. If the CSP only consists of binary constraints, arcs connecting nodes with variable are enough to represent constraints.
2. If the CSP also consists of high-order constraints, we use square nodes to represent the constraints and connect them with the involved nodes with variable using arcs.

Example 1.1.3. Using the map colouring problem in Example 1.1.1, we have the following constraint graph:



Example 1.1.4. Using the cryptarithmetic puzzle in Example 1.1.2, we have the following constraint graph:



2 Solving the Constraint Satisfaction Problem using Search

Using constraint graph, we can format the search formulation of CSPs.

State space: Each state defined by the values assigned to the variables

Initial state: Empty assignment: {}

Successor function: Assign a value to an unassigned variable

Goal test: Current assignment is complete and satisfies all constraint

We can try Depth-first search (DFS) to solve this problem.

Example 2.0.1. We use the map colouring problem in Example 1.1.1. There are 10 variables.

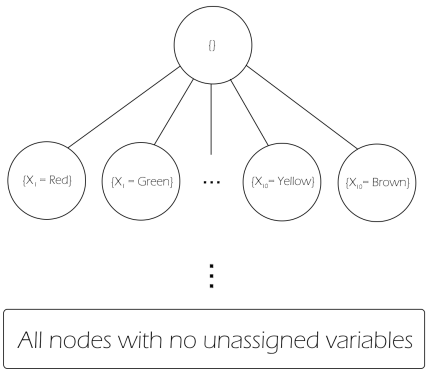
Number of values:
$d = 4$

Number of unassigned variables at Level i:
$n = 11 - i$

Branch factor from Level i to Level $i + 1$:
Branch factor $b = dn = 4(11 - i)$

Number of nodes at Level $i > 2$:
$4^{i-1}(11-1)\cdots(11-i+1) = \frac{4^{i-1}10!}{(11-i)!}$

Height of the tree:
Each arc down assign a variable. Height $h = 11$



As you can see, it is highly inefficient. We can find some issues regarding the search.

1. Variables assignments are communicate.
   E.g. $\{X_1 = \text{Red}, X_2 = \text{Green}\}$ same as $\{X_2 = \text{Green}, X_1 = \text{Red}\}$
   Modification: Consider assignments to a single variable at each node.
   Branching factor: $b = d$ Number of nodes at Level i: $d^{i=1}$

2. Each variable can be assigned with any value
   E.g. $\{X_1 = \text{Red}, X_2 = \text{Red}\}$
   Modification: Consider only values which do not conflict with previous assignments.

After these two modification, we have a new type of search called **backtracking search**.

Definition 2.1. Backtracking search is a searching algorithm that involves trying different options and undoing then if they lead to a dead end. The idea is that:

1. One variable at a time
2. Check constraints as you go
3. Backtrack when a variable has no legal values left to assign.

Algorithm 1 Backtracking algorithm

RECURSIVE-BACKTRACKING(assignment, csp)

Input: assignment, csp
Output: solution or failure

```
if assignment is complete then
    return assignment
end if
var ← SELECT-UNASSIGNED-VARIABLE(VARIABLE[csp], assignment, csp)
for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
    if value is consistent with assignment given CONSTRAINT[csp] then
        add {var = value} to assignment
        result ← RECURSIVE-BACKTRACKING(assignment, csp)
        if result ≠ failure then
            return result
        end if
        remove {var = value} from assignment
    end if
end for
return failure
```

First Call: RECURSIVE-BACKTRACKING({}, csp)

The above backtracking algorithm has the following steps:

1. Use a method to select unassigned variable. (SELECT-UNASSIGNED-VARIABLE($\cdots$))
2. Use a method to select value. (ORDER-DOMAIN-VALUES($\cdots$))
3. Check if the variable assigned with the value is consistent with the constraints.
4. Assign the variable with the value and perform the algorithm again.
5. If a solution exists with that assignment, we return the assignment. If not, we assign the variable with the next value.
6. Return failure if there are no available values.

However, how do we determine which variable or value we should select first?

3 Ordering

How do we choose a variable? We may choose the one with the least remaining legal values.

Definition 3.1. Minimum Remaining Values (MRV) is an ordering method of choosing a variable by choosing the variable with the fewest legal values left in its domain.

We can use a tie-breaking strategy. E.g. The first variable is chosen first.

Example 3.1.1. We use the cryptarithmetic puzzle in Example 1.1.2. We have the initial possible values.

Variable	G	B	A	M	S	L	E
Possible values	$\{1\}$	$\{1, \cdots, 9\}$	$\{0, \cdots, 9\}$	$\{0, \cdots, 9\}$	$\{0, \cdots, 9\}$	$\{0, \cdots, 9\}$	$\{0, \cdots, 9\}$

By MRV, we can choose G since it has the minimum remaining values. We can assign the only value in the domain 1 to G.

$$\begin{array}{r} \texttt{B A S E} \\ +\ \texttt{B A L L} \\ \hline \texttt{G A M E S} \end{array} \Longrightarrow \begin{array}{r} \texttt{B A S E} \\ +\ \texttt{B A L L} \\ \hline \texttt{1 A M E S} \end{array}$$

Why do we do this? If the most constrained variable is left unassigned, we may easily lead to backtracking later on as the number of remaining legal values decreases in the domain of the variable.
How do we choose a value for a variable? We may choose the one that can be assigned to the fewest variables.

Definition 3.2. Least Constraining Value (LCV) is an ordering method of choosing a value by choosing the value that rules out the fewest values in the remaining variables.

We can use a tie-breaking strategy. E.g. The smallest value is chosen first.

Example 3.2.1. We again use the cryptarithmetic puzzle in Example 1.1.2. Consider the case when we have already assigned $G = 1$ ($C_3 = 1$), $S = 8$, $L = 5$ and $E = 3$ ($C_1 = 0$, $C_2 = 1$). We check the constraints and get the following possible values for each variables:

Variable	G	B	A	M	S	L	E
Possible values	1	$\{6, 7, 9\}$	$\{0, 2, 4, 6, 7, 9\}$	$\{7, 9\}$	8	5	3

We want to assign a value for A, which one should we choose? We check the number of values removed in the domain using the constraints.

Alldiff(G, B, A, M, S, L, E)
$2 \times B + C_3 = A + 10$
$2 \times A + 1 = M + 10 \times C_3$

A	0	2	4	6	7	9
No. of removed values in B	3	2	2	3	3	3
No. of removed values in M	2	2	1	2	2	2

By LCV, since 4 removes the least number of values of the domains, we assign 4 to A.

$$\begin{array}{r} \texttt{B A 8 3} \\ +\ \texttt{B A 5 5} \\ \hline \texttt{1 A M 3 8} \end{array} \Longrightarrow \begin{array}{r} \texttt{B 4 8 3} \\ +\ \texttt{B 4 5 5} \\ \hline \texttt{1 4 M 3 8} \end{array}$$

Why do we do this? This allows for more freedom of assignments to the other unassigned variables and potentially lead to fewer backtracking. Usually, we use both MRV and LCV at the same time.

4 Filtering

In the above example, we check **all** the values in the domain whether it violates the constraint after we have assigned the value to the variable. This is highly inefficient as the domain doesn't change. How about we keep track of the domain of each value and update the domain as we check the constraint every time?

Definition 4.1. Forward checking is a filtering method of checking the domain of unassigned variables by crossing off values that violate constraints after adding an existing assignment.

Example 4.1.1. We use the cryptarithmetic puzzle in Example 3.1.1. Originally, the domain is listed as follows:

Variable	G	B	A	M	S	L	E
Domain	$\{1\}$	$\{1,2,\cdots,9\}$	$\{0,1,\cdots,9\}$	$\{0,1,\cdots,9\}$	$\{0,1,\cdots,9\}$	$\{0,1,\cdots,9\}$	$\{0,1,\cdots,9\}$

We list out the constraints:

$$\begin{cases} E+L=S+10C_1 \\ S+L+C_1=E+10C_2 \\ 2A+C_2=M+10C_3 \\ 2B+C_3=A+10C_4 \\ G=C_4 \end{cases}$$

After assigning $G=1$. The constraints become:

$$\begin{cases} E+L=S+10C_1 \\ S+L+C_1=E+10C_2 \\ 2A+C_2=M+10C_3 \\ 2B+C_3=A+10C_4 \\ 1=C_4 \end{cases} \implies \begin{cases} E+L=S+10C_1 \\ S+L+C_1=E+10C_2 \\ 2A+C_2=M+10C_3 \\ 2B+C_3=A+10 \end{cases}$$

We can now update the domain of the values using the constraints.

Variable	G	B	A	M	S	L	E
Domain	1	$\{5,6,7,8,9\}$	$\{0,2,\cdots,9\}$	$\{0,2,\cdots,9\}$	$\{0,2,\cdots,9\}$	$\{0,2,\cdots,9\}$	$\{0,2,\cdots,9\}$

By forward checking, we have reduced the size of the domain. Can we do better? Can we detect the failure even sooner, possibly involve those unassigned variables that do not have direct constraint with the last assignment?

Definition 4.2. Constraint Propagation is a filtering method which involves reason further from constraint to constraint among unassigned variables.

How do we use this method? We can use arc consistency to propagate constraint information among unassigned variables.

Definition 4.3. An arc $X \to Y$ is **consistent** if for every $x \in X$ there is some $y \in Y$ which could be assigned without violating a constraint.

Example 4.3.1. We continue the Example 4.1.1. Assume that we have assigned $M = 2$. After forward checking, the domain is listed as follows:

Variable	G	B	A	M	S	L	E
Domain	1	$\{5, 6, 7, 8, 9\}$	$\{1, 6\}$	2	$\{0, 3, \cdots, 9\}$	$\{0, 3, \cdots, 9\}$	$\{0, 3, \cdots, 9\}$

We list all the constraints.

$$\begin{cases} E + L = S + 10C_1 \\ S + L + C_1 = E + 10C_2 \\ 2A + C_2 = 2 + 10C_3 \\ 2B + C_3 = A + 10 \end{cases} \implies \begin{cases} E + L = S + 10C_1 \\ S + L + C_1 = E \\ 2A = 2 + 10C_3 \\ 2B + C_3 = A + 10 \end{cases}$$

We can use the arc consistency to check the values by looking at constraint 4. Start with $B \to A$.

1. If $B = 5$, then A can be assigned with 1. However, this violates constraint 3 since $C_3 = 1$ by constraint 4. Therefore, A cannot be assigned with anything.
2. If $B = 8$, then A can be assigned with 6. However, this violates constraint 3 since $C_3 = 0$ by constraint 4. Therefore, A cannot be assigned with anything.
3. If $B = 6, 7, 9$, then A cannot be assigned with anything.

From this, we can update the domain as below:

Variable	G	B	A	M	S	L	E
Domain	1	$\{\}$	$\{1, 6\}$	2	$\{0, 3, \cdots, 9\}$	$\{0, 3, \cdots, 9\}$	$\{0, 3, \cdots, 9\}$

However, this means that B cannot be assigned with anything. Therefore, we cannot assign $M = 2$.

Example 4.3.2. Continuing the last example. What if we assign $M = 8$ instead? After forward checking, the domain is listed as follows:

Variable	G	B	A	M	S	L	E
Domain	1	$\{5, 6, 7, 9\}$	$\{4, 9\}$	8	$\{0, 2, \cdots, 7, 9\}$	$\{0, 2, \cdots, 7, 9\}$	$\{0, 2, \cdots, 7, 9\}$

We list all the constraints.

$$\begin{cases} E + L = S + 10C_1 \\ S + L + C_1 = E + 10C_2 \\ 2A + C_2 = 8 + 10C_3 \\ 2B + C_3 = A + 10 \end{cases} \implies \begin{cases} E + L = S + 10C_1 \\ S + L + C_1 = E \\ 2A = 8 + 10C_3 \\ 2B + C_3 = A + 10 \end{cases}$$

We can use the arc consistency to check the values by looking at constraint 4. Start with $B \to A$.

1. If $B = 7$, then A can be assigned with 4.
2. If $B = 5, 6, 9$, then A cannot be assigned with anything.

From this, we can update the domain as below:

Variable	G	B	A	M	S	L	E
Domain	1	$\{7\}$	$\{4, 9\}$	8	$\{0, 2, \cdots, 7, 9\}$	$\{0, 2, \cdots, 7, 9\}$	$\{0, 2, \cdots, 7, 9\}$

Now we check $A \to B$.

1. If $A = 4$, then B can be assigned with 7.
2. If $A = 9$, then B cannot be assigned with anything.

From this, we can update the domain as below:

Variable	G	B	A	M	S	L	E
Domain	1	$\{7\}$	$\{4\}$	8	$\{0, 2, \cdots, 7, 9\}$	$\{0, 2, \cdots, 7, 9\}$	$\{0, 2, \cdots, 7, 9\}$

Note that after you have changed the domain of a variable, you need to check the arcs connected to that variable again.