Constraint Satisfaction Problem

Constraint Satisfaction Problem (CSP)

- A problem is solved when each variable has a value that satisfies all the constraints on the variable. A problem described this way is called a **constraint satisfaction problem**, or CSP.

- The main idea is to eliminate large portions of the search space all at once by identifying variable/value combinations that violate the constraints.

Definition of CSP

- Constraint satisfaction problem consists of three components, X, D, and C:
  - X is a set of variables, $\{X_1,...,X_n,\}$.
  - D is a set of domains, $\{D_1,...,D_n,\}$, one for each variable.
  - C is a set of constraints that specify allowable combinations of values.

- A domain, D, consists of a set of allowable values, $\{v_1,..., v_n\}$, for variable X;.
  - For example, a Boolean variable would have the domain {true,false}.

- Each constraint C_j consists of a pair (scope, rel),
  - where scope is a tuple of variables that participate in the constraint
  - rel is a relation that defines the values that those variables can take on.

- A relation can be represented as an explicit set of all tuples of values that satisfy the constraint, or as a function that can compute whether a tuple is a member of the relation.

  - For example, if X_1 and X_2 both have the domain {1,2,3}, then the constraint saying that X_1; must be greater than X_2 can be written as,

$((X_1,X_2),\{(3,1),(3,2),(2,1)\})$ or as $((X_1,X_2), X_1 > X_2)$.

- CSPs deal with assignments of values to variables, $\{X_i = v_i, X_j=v_j,...\}$.

- An assignment that does not violate any constraints is called a **consistent or legal assignment**.

- A **complete assignment** is one in which every variable is assigned a value, and a **solution** to a CSP is a consistent, complete assignment.

- A **partial assignment** is one that leaves some variables unassigned, and a partial solution is a partial assignment that is **consistent**.

Example problem: Map coloring

- A map of Australia showing each of its states and territories.
- We are given the task of coloring each region either red, green, or blue in such a way that no two neighboring regions have the same color.

Example problem: Map coloring

- We define the variables to be the regions:

$$X = \{WA, NT, Q, NSW, V, SA, T\}.$$

- The domain of every variable is the set

$$D_i = \{red, green, blue\}.$$

- The constraints require neighboring regions to have distinct colors.
- Since there are nine places where regions border, there are nine constraints:

$$C = \{SA \neq WA,\ SA \neq NT,\ SA \neq Q,\ SA \neq NSW,\ SA \neq V,$$
$$WA \neq NT,\ NT \neq Q,\ Q \neq NSW,\ NSW \neq V\}.$$

Example problem: Map coloring

- Here we using arc abbreviations; SA ≠ WA is a shortcut for ((SA,WA),SA ≠ WA),

- Where SA ≠ WA can be fully enumerated in turn as

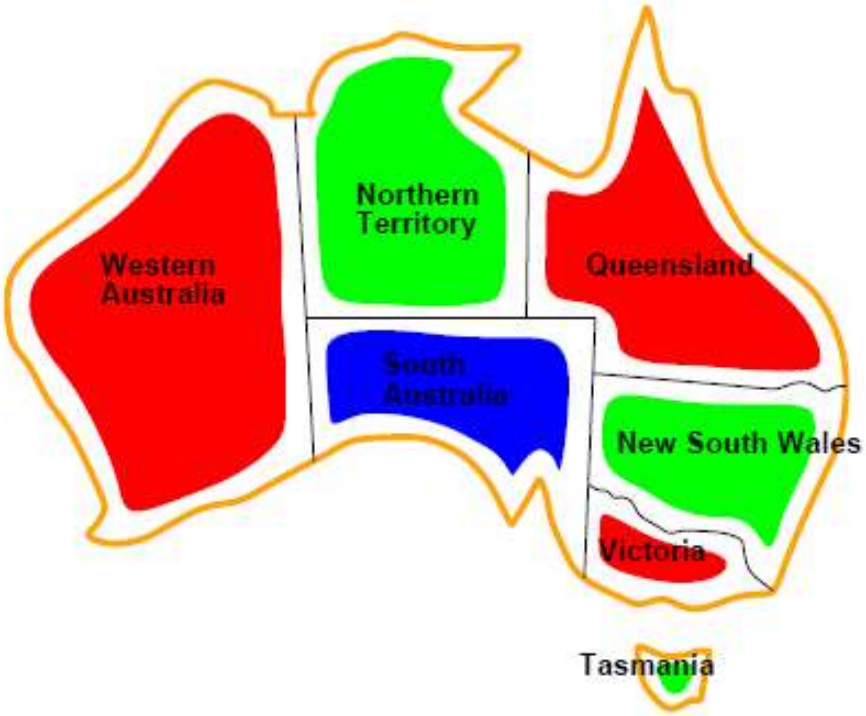
{(red, green), (red. blue), (green, red), (green, blue), (blue, red), (blue, green) }.

- Solutions to this problem?

Example problem: Map coloring

- There are many possible solutions to this problem, such as

{WA =red, NT=green, Q= red, NSW = green, V = red, SA=blue, T = red }.

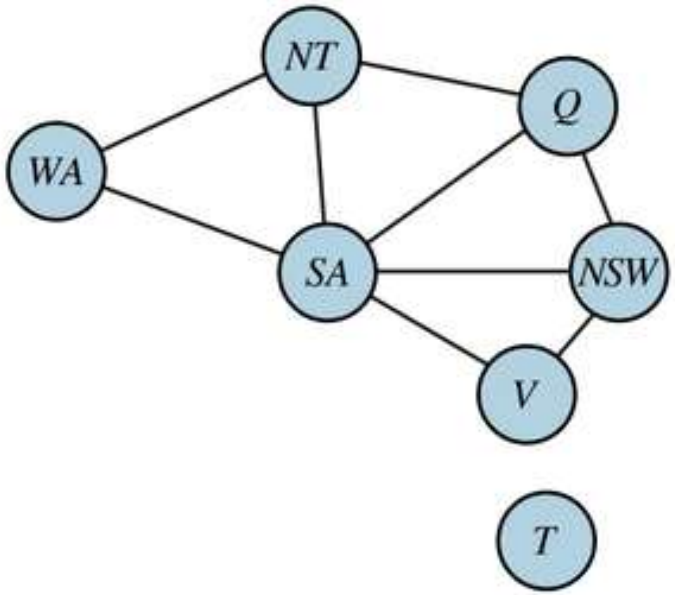


Solutions are assignments satisfying all constraints, e.g.,

$\{WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green\}$

Example problem: Map coloring

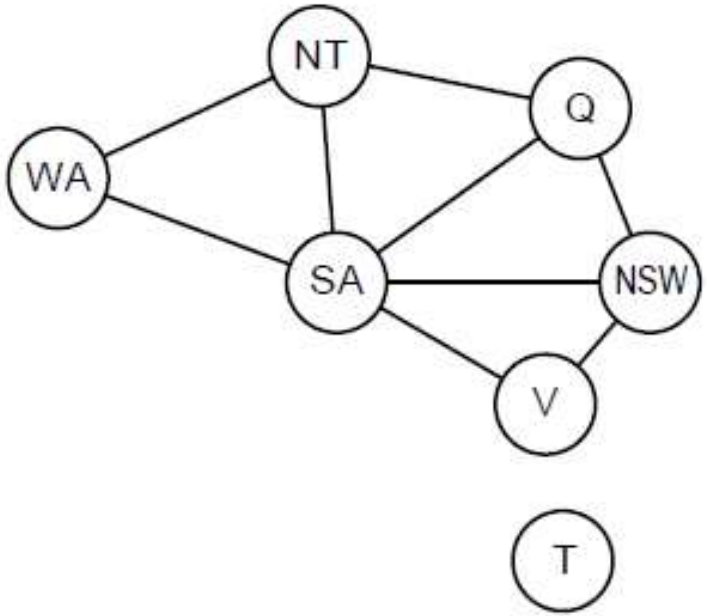
- It can be helpful to visualize a CSP as a constraint graph, as shown in Figure.
- The nodes of the graph correspond to variables of the problem, and an edge connects any two variables that participate in a constraint.



Constraint graph

Binary CSP: each constraint relates at most two variables

Constraint graph: nodes are variables, arcs show constraints



General-purpose CSP algorithms use the graph structure
to speed up search. E.g., Tasmania is an independent subproblem!

Types of CSP

Discrete variables

- finite domains; size d $\Rightarrow$ $O(d^n)$ complete assignments
  - ♢ e.g., Boolean CSPs, incl. Boolean satisfiability (NP-complete)
- infinite domains (integers, strings, etc.)
  - ♢ e.g., job scheduling, variables are start/end days for each job
  - ♢ need a constraint language, e.g., $StartJob_1 + 5 \leq StartJob_3$
  - ♢ linear constraints solvable, nonlinear undecidable

Continuous variables

- ♢ e.g., start/end times for Hubble Telescope observations
- ♢ linear constraints solvable in poly time by LP methods

Varieties of constraints

Unary constraints involve a single variable,
e.g., $SA \neq green$

Binary constraints involve pairs of variables,
e.g., $SA \neq WA$

Higher-order constraints involve 3 or more variables,
e.g., cryptarithmetic column constraints

Preferences (soft constraints), e.g., red is better than $green$
often representable by a cost for each variable assignment
$\rightarrow$ constrained optimization problems

Cryptarithmetic Problem

- Cryptarithmetic Problem is a type of constraint satisfaction problem where the game is about digits and its unique replacement either with alphabets or other symbols.

- In **cryptarithmetic problem,** the digits (0-9) get substituted by some possible alphabets or symbols.

- The task in cryptarithmetic problem is to substitute each digit with an alphabet to get the result arithmetically correct.

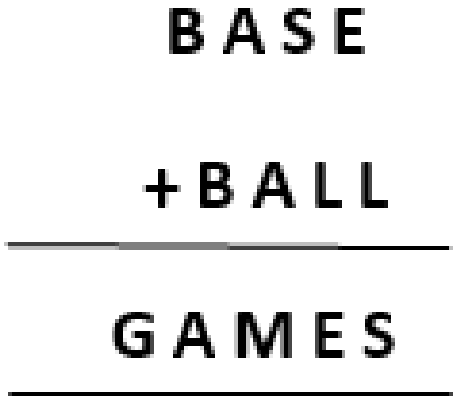
Rules and constraints

- There should be a unique digit to be replaced with a unique alphabet.
- The result should satisfy the predefined arithmetic rules, i.e., 2+2 =4, nothing else.
- Digits should be from **0-9** only.
- The problem can be solved from both sides, i.e., **lefthand side (L.H.S), or righthand side (R.H.S)**

Example-1

```
  BASE
+BALL
-----
 GAMES
-----
```

Example-1



Variables: {B,A,S,E,L,G,M}
Domain:{0,1,2,3,4,5,6,7,8,9}
Constraint: AllDiffz(B,A,S,E,L,G,M)

Example-1

```
  B A S E
+ B A L L
---------
G A M E S
---------
```

B	5
B	5
G A	10
G	1
A	0

Example-1

```
  B A S E
+ B A L L   _
-----------
G A M E S
-----------
```

B	6
B	6
G A	12
G	1
A	2

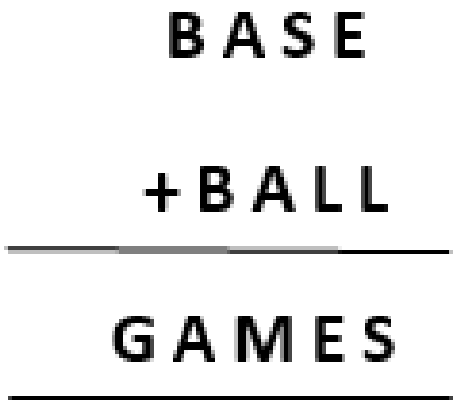
Example-1

Cr	0	Cr	0
B	6	A	2
B	6	A	2
G A	12	M	4
G	1		
A	2		

Example-1

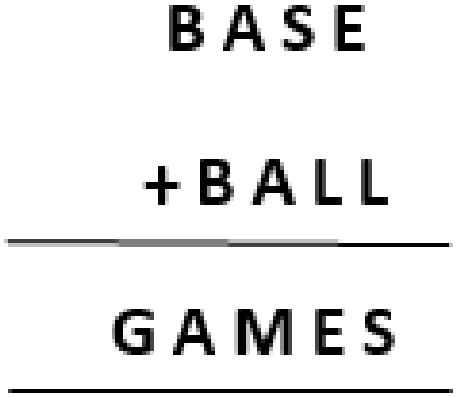
Cr	0	Cr	0	Cr	
B	6	A	2	S	3
B	6	A	2	L	5
G A	12	M	4	E	8
G	1				
A	2				

Example-1



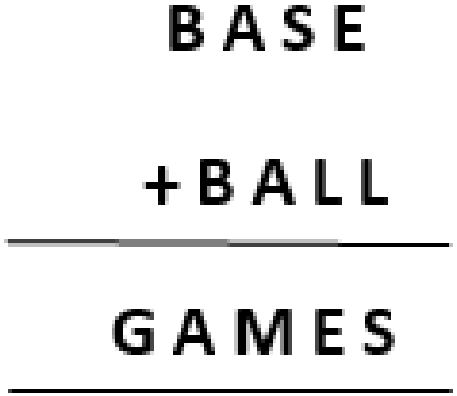
Cr	0	Cr	0	Cr		Cr	
B	6	A	2	S	3	E	8
B	6	A	2	L	5	L	5
G A	12	M	4	E	8	S	
G	1						
A	2						

Example-1 Solution



Cr	0	Cr	0	Cr		Cr	
B	**7**	A		S		E	
B	**7**	A		L		L	
G A	**14**	M		E		S	
G	**1**						
A	**4**						

Example-1 Solution

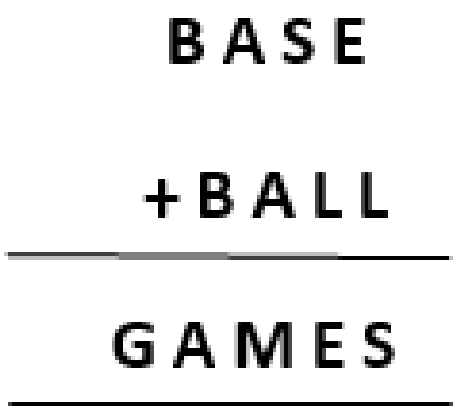


Cr	0	Cr	0	Cr		Cr	
B	**7**	A	**4**	S		E	
B	**7**	A	**4**	L		L	
G A	**14**	M	**8**	E		S	
G	**1**						
A	**4**						

Example-1 Solution

Cr	0	Cr	*1*	Cr		Cr	
B	**7**	A	**4**	S	**8**	E	
B	**7**	A	**4**	L	**5**	L	
G A	**14**	M	**9**	E	**3**	S	
G	**1**						
A	**4**						

Example-1 Solution



Cr	0	Cr	*1*	Cr	0	Cr	0
B	**7**	A	**4**	S	**8**	E	**3**
B	**7**	A	**4**	L	**5**	L	**5**
G A	**14**	M	**9**	E	**3**	S	**8**
G	**1**						
A	**4**						

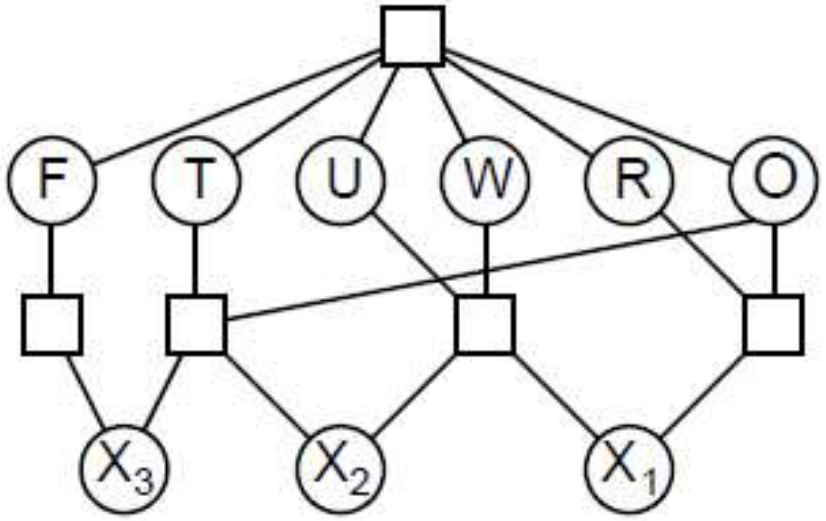
B	7
A	4
S	8
E	3
L	5
G	1
M	9

```
  S E N D
+ M O R E
---------
M O N E Y
---------
```

Cr		Cr		Cr		Cr	
S		E		N		D	
M		O		R		E	
MO		N		E		Y	
M							
O							

Example: Cryptarithmetic

$$\begin{array}{r} \ T\ W\ O \\ +\ T\ W\ O \\ \hline F\ O\ U\ R \end{array}$$



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)

$O + O = R + 10 \cdot X_1$, etc.

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

Cr	*0*	*Cr*	*0*	*Cr*	*0*
T	**7**	W	**3**	O	**4**
T	**7**	W	**3**	O	**4**
FO	**14**	U	**6**	R	**8**
F	**1**				
O	**4**				

Real-world CSPs

Assignment problems
    e.g., who teaches what class

Timetabling problems
    e.g., which class is offered when and where?

Hardware configuration

Spreadsheets

Transportation scheduling

Factory scheduling

Floorplanning

Notice that many real-world problems involve real-valued variables

Standard search formulation (incremental)

Let's start with the straightforward, dumb approach, then fix it

States are defined by the values assigned so far

- Initial state: the empty assignment, $\{\}$
- Successor function: assign a value to an unassigned variable that does not conflict with current assignment.
  $\Rightarrow$ fail if no legal assignments (not fixable!)
- Goal test: the current assignment is complete

1) This is the same for all CSPs! 
2) Every solution appears at depth n with n variables
   $\Rightarrow$ use depth-first search
3) Path is irrelevant, so can also use complete-state formulation
4) $b = (n - \ell)d$ at depth ℓ, hence $n!d^n$ leaves!!!!