Constraint Satisfaction Problem

CSP – Constraint Satisfaction Problem

Standard Search Problem >>

- Searches a space of states for the solution.
- Heuristics are domain specific.
- Each state is atomic/indivisible i.e. a black box with no internal structure.

CSP >>

- CSP uses a factored representation for each state: a set of variables, each of which has a value.
- A problem is solved when each variable has a value that satisfies all the constraints in the variable.
- A problem described this way is called a *constraint satisfaction problem*.

Advantages

- Uses general-purpose rather than problem-specific heuristics to enable the solution of complex problems.
- Represents each state as a set of variables and takes advantage of the structure of states.
- Eliminates large portions of the search space all at once by identifying variable/value combinations that violate the constraints.

CSP >> Formal Definition

A constraint satisfaction problem consists of three components, *X*, *D*, and *C*:

- *X* is the set of variables, $\{ X_1, \ldots, X_n \}$
- *D* is a set of domains, $\{ D_1, \ldots, D_n \}$, one for each variable.
- *C* is a set of constraints that specify allowable combinations of values.

Here,

- Each domain D_i consists of a set of allowable values, $\{ v_1, \ldots, v_k \}$ for each variable X_i
- Each constraint C_i consists of a pair $\langle scope, rel \rangle$,
  where *scope* is a tuple of variables that participate in the constraint
  and *rel* is a relation that defines the values that those variables can take on.
- A relation can be represented as an *explicit* list of all tuples of values that satisfy the constraint, or as an *abstract* relation. For example,

$$\langle (X_1, X_2), [(A, B), (B, A)] \rangle \text{ or } \langle (X_1, X_2), X_1 \mathrel{!}= X_2 \rangle$$

CSP >> Solution

- Each state in a CSP is defined by an assignment of values to some or all of the variables.

$$\{ X_i = v_i , X_j = v_j , \dots \}$$

- 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 **partial assignment** is one that assigns values to only some of the variables.

- A **solution** to a CSP is consistent and complete assignment.

Advantages of formulating a problem as a CSP

- CSPs yield natural representation for a wide variety of problems.
- CSP solvers can be faster than state-space searchers because the CSP solver can quickly eliminate large swatches of the search space.
- With CSP, once we find out that a partial assignment is not a solution, we can immediately discard further refinements of the partial assignment.
- We can see why a assignment is not a solution – which variables violate a constraint – we can focus attention on the variables that matter.

CSP >> Problem Formulation

Map Coloring Problem >>

your task is to color each region either *red, green, or blue* in such a way that no neighboring regions have the same color.

CSP Formulation

Each region is a variable, X = { WA, NT, Q, NSW, V, SA, T }

Domain of each variable, D_i = { red, green, blue }

Set of constraints,
C = { SA ≠ WA, SA ≠ NT, SA ≠ Q, SA ≠ NSW, SA ≠ V,
WA ≠ NT, NT ≠ Q, Q ≠ NSW, NSW ≠ V }

Here, SA ≠ WA is a shortcut for $\langle scope, rel \rangle = \langle (SA, WA),\ SA \neq WA \rangle$

CSP >> Possible Solution

There are many possible solutions to this problem,

Such as

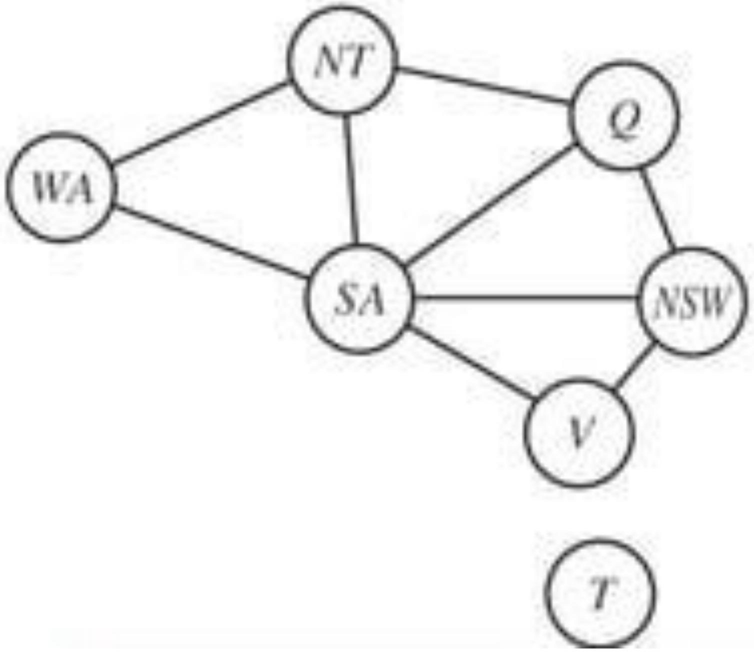
{ WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green }

CSP >> Constraint Graph

Binary CSP >> each constraint relates at most two variables.

Constraint Graph >>

- Nodes correspond to the variables of the problem.
- An edge connects any two variables that participate in a constraint.



CSP >> Job-shop Scheduling

Let's consider a small part(wheel installation) of the car assembly, consisting of 15 tasks.

- Install axles (front and back) – requires 10 mins to install
- Affix all four wheels (right and left, front and back) – takes 1 mins
- Tighten nuts for each wheel – takes 2 mins
- Affix hubcaps and – requires 1 mins
- Inspect the final assembly – takes 3 mins

And get the whole assembly done in 30 mins

Inspect

CSP >> Job-shop Scheduling

We can represent the tasks with 15 variables:

X = { $Axle_F$, $Axle_B$, $Wheel_{RF}$, $Wheel_{LF}$, $Wheel_{RB}$, $Wheel_{LB}$, $Nuts_{RF}$, $Nuts_{LF}$, $Nuts_{RB}$, $Nuts_{LB}$, Cap_{RF}, Cap_{LF}, Cap_{RB}, Cap_{LB}, Inspect }

Precedence constraints >> whenever a task T_1 must occur before task T_2 and task T_1 takes duration d_1 to complete, then we can represent the constraint as

$$T_1 + d_1 <= T_2$$

So the constraints are,
C = {

$Axle_F + 10 <= Wheel_{RF}$, $Axle_F + 10 <= Wheel_{LF}$, $Axle_B + 10 <= Wheel_{RB}$, $Axle_B + 10 <= Wheel_{LB}$,
$Wheel_{RF} + 1 <= Nuts_{RF}$, $Wheel_{LF} + 1 <= Nuts_{LF}$, $Wheel_{RB} + 1 <= Nuts_{RB}$, $Wheel_{LB} + 1 <= Nuts_{LB}$,
$Nuts_{RF} + 2 <= Cap_{RF}$, $Nuts_{LF} + 2 <= Cap_{LF}$, $Nuts_{RB} + 2 <= Cap_{RB}$, $Nuts_{LB} + 2 <= Cap_{LB}$,
$Cap_{RF} + 1 <=$ Inspect, $Cap_{LF} + 1 <=$ Inspect, $Cap_{RB} + 1 <=$ Inspect, $Cap_{LB} + 1 <=$ Inspect,

$Axle_F + 10 <=$ Inspect, $Axle_B + 10 <=$ Inspect,
$Wheel_{RF} + 1 <=$ Inspect , $Wheel_{LF} + 1 <=$ Inspect , $Wheel_{RB} + 1 <=$ Inspect , $Wheel_{LB} + 1 <=$ Inspect ,
$Nuts_{RF} + 2 <=$ Inspect , $Nuts_{LF} + 2 <=$ Inspect , $Nuts_{RB} + 2 <=$ Inspect , $Nuts_{LB} + 2 <=$ Inspect

}
As inspection takes 3 mins to complete and the total task need to complete within 30 minutes,
So domain of each variable, D_i = { 1, 2, 3, ... , 27 }