Constraint Satisfaction Problems

Michael Rose

In which we see how treating states as more than just little black boxes leads to the invention of a range of powerful new search methods and a deeper understanding of problem structure and complexity.

In this chapter, we solve problems in which each state has a **factored** representation: 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 on the variable. A problem described this way is called a **constraint satisfaction problem**.

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

6.1 | Defining Constraint Satisfaction Problems

A constraint satisfaction problem consists of 3 components, X, D, and C:

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

To solve a CSP, we need to define a srare space and the notion of a solution. Each state in a CSP is defined by an **assignment** of values to some or all of the variables, $\{X_i = v_i, X_j = v_j, ...\}$. 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, and a solution to a CSP is a consistant, complete assignment. A **partial assignment** is one that assigns values to only some of the variables.

6.1.3 | Variations on the CSP Formalism

The simplest kinds of CSP involve variables that have discrete, finite domains.

If we have a discrete domain that is infinite, we can no longer enumerate the options. Instead, we must use a **constraint language** that understands constraints like $T_1 + d_1 \leq T_2$ directly without enumerating all pairs of allowable values.

Special solution algorithms exist for linear constraints, but it can be shown that no algorithm exists for solving general nonlinear constraints on integer variables.

CSPs with **continuous domains** are common in the real world. The best known categories of these are called **linear programming** problems, where constraints must be linear equalities or inequalities.

The simplest type of constraint is called a **unary constraint**, which restricts our value to that of a single variable. Similarly, a binary constraint involves 2 values. We can also assert that values are between values, like an inequality. A constraint involving an arbitrary number (but not necessarily all of) variables is called a **global constraint**.

Constraints needn't be black and white. Many real-world CSPs include **preference constraints**, indicating which solutions are preferred. With these in a problem formulation, CSPs with preferences can be solved with optimization search methods, either path-based or local. This is called a **Constraint Optimization Problem**. Linear Programming problems do this kind of optimization.

6.2 | Constraint Propagation: Inference in CSPs

In regular state space algorithms, we can only perform one operation: search. In CSPs, we can also do a type of inference called **constraint propagation**. This is when we use constraints to reduce the number of legal values for a variable. The key idea is **local consistency**. If we treat each variable as a node in a graph, and each binary constraint as an arc, then the process of enforcing local consistency in each part of the graph causes inconsistent values to be eliminated throughout the graph.

There are different types of local consistency:

6.2.1 | Node Consistency

A single variable (corresponding to a node in the CSP network) is node-consistent if all the values in the variable's domain satisfy the variable's unary constraints.

6.2.2 | Arc Consistency

A variable in a CSP is arc-consistent if every value in its domain satisfies the variable's binary constraints. More formally, X_i is arc-consistent with respect to another variable X_j is for every value in the current domain D_i there is some value in the domain D_j that satisfies the binary constraint on the arc (X_i, X_j).

It is possible to extend the notion of arc-consistency from binary to n-ary constraints.

6.2.3 | Path Consistency

6.2.4 | K-Consistency