6 CONSTRAINT SATISFACTION PROBLEMS

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.

Chapters 3 and 4 explored the idea that problems can be solved by searching in a space of **states**. These states can be evaluated by domain-specific heuristics and tested to see whether they are goal states. From the point of view of the search algorithm, however, each state is atomic, or indivisible—a black box with no internal structure.

This chapter describes a way to solve a wide variety of problems more efficiently. We use 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 on the variable. A problem described this way is called a **constraint satisfaction problem**, or CSP.

CONSTRAINT SATISFACTION PROBLEM

CSP search algorithms take advantage of the structure of states and use *general-purpose* rather than *problem-specific* heuristics to enable the solution of complex problems. 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 three components, X, D, and C:

X is a 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.

Each domain D_i consists of a set of allowable values, $\{v_1, \ldots, v_k\}$ for 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 that supports two operations: testing if a tuple is a member of the relation and enumerating the members of the relation. For example, if X_1 and X_2 both have the domain {A,B}, then the constraint saying the two variables must have different values can be written as $\langle (X_1, X_2), [(A, B), (B, A)] \rangle$ or as $\langle (X_1, X_2), X_1 \neq X_2 \rangle$.

To solve a CSP, we need to define a state 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, $\{X_i = v_i, X_j = v_j, \ldots\}$. 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 consistent, complete assignment. A **partial assignment** is one that assigns values to only some of the variables.

ASSIGNMENT
CONSISTENT
COMPLETE ASSIGNMENT
SOLUTION
PARTIAL ASSIGNMENT

6.1.1 Example problem: Map coloring

Suppose that, having tired of Romania, we are looking at a map of Australia showing each of its states and territories (Figure 6.1(a)). We are given the task of coloring each region either red, green, or blue in such a way that no neighboring regions have the same color. To formulate this as a CSP, we define the variables to be the regions

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

The domain of each variable is the set $D_i = \{\mathit{red}, \mathit{green}, \mathit{blue}\}$. The constraints require neighboring regions to have distinct colors. Since there are nine places where regions border, there are nine constraints:

$$\begin{aligned} C = \{&\mathit{SA} \neq \mathit{WA}, \mathit{SA} \neq \mathit{NT}, \mathit{SA} \neq Q, \mathit{SA} \neq \mathit{NSW}, \mathit{SA} \neq V, \\ &\mathit{WA} \neq \mathit{NT}, \mathit{NT} \neq Q, Q \neq \mathit{NSW}, \mathit{NSW} \neq V\} . \end{aligned}$$

Here we are using abbreviations; $\mathit{SA} \neq \mathit{WA}$ is a shortcut for $\langle (\mathit{SA}, \mathit{WA}), \mathit{SA} \neq \mathit{WA} \rangle$, where $\mathit{SA} \neq \mathit{WA}$ can be fully enumerated in turn as

$$\{(\mathit{red}, \mathit{green}), (\mathit{red}, \mathit{blue}), (\mathit{green}, \mathit{red}), (\mathit{green}, \mathit{blue}), (\mathit{blue}, \mathit{red}), (\mathit{blue}, \mathit{green})\} .$$

There are many possible solutions to this problem, such as

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

CONSTRAINT GRAPH

It can be helpful to visualize a CSP as a **constraint graph**, as shown in Figure 6.1(b). The nodes of the graph correspond to variables of the problem, and a link connects any two variables that participate in a constraint.

Why formulate a problem as a CSP? One reason is that the CSPs yield a natural representation for a wide variety of problems; if you already have a CSP-solving system, it is often easier to solve a problem using it than to design a custom solution using another search technique. In addition, CSP solvers can be faster than state-space searchers because the CSP solver can quickly eliminate large swatches of the search space. For example, once we have chosen $\{\mathit{SA} = \mathit{blue}\}$ in the Australia problem, we can conclude that none of the five neighboring variables can take on the value blue. Without taking advantage of constraint propagation, a search procedure would have to consider $3^5 = 243$ assignments for the five neighboring variables; with constraint propagation we never have to consider blue as a value, so we have only $2^5 = 32$ assignments to look at, a reduction of 87%.

In regular state-space search we can only ask: is this specific state a goal? No? What about this one? With CSPs, once we find out that a partial assignment is not a solution, we can

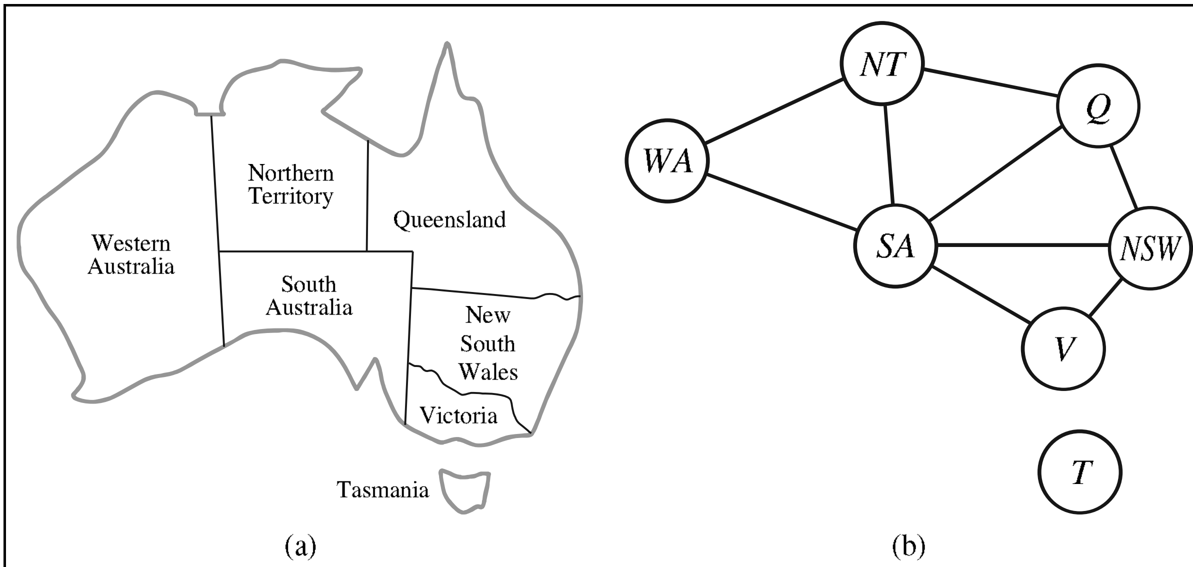

Figure 6.1 (a) The principal states and territories of Australia. Coloring this map can be viewed as a constraint satisfaction problem (CSP). The goal is to assign colors to each region so that no neighboring regions have the same color. (b) The map-coloring problem represented as a constraint graph.

immediately discard further refinements of the partial assignment. Furthermore, we can see *why* the assignment is not a solution—we see which variables violate a constraint—so we can focus attention on the variables that matter. As a result, many problems that are intractable for regular state-space search can be solved quickly when formulated as a CSP.

6.1.2 Example problem: Job-shop scheduling

Factories have the problem of scheduling a day's worth of jobs, subject to various constraints. In practice, many of these problems are solved with CSP techniques. Consider the problem of scheduling the assembly of a car. The whole job is composed of tasks, and we can model each task as a variable, where the value of each variable is the time that the task starts, expressed as an integer number of minutes. Constraints can assert that one task must occur before another—for example, a wheel must be installed before the hubcap is put on—and that only so many tasks can go on at once. Constraints can also specify that a task takes a certain amount of time to complete.

We consider a small part of the car assembly, consisting of 15 tasks: install axles (front and back), affix all four wheels (right and left, front and back), tighten nuts for each wheel, affix hubcaps, and inspect the final assembly. We can represent the tasks with 15 variables:

$$\begin{aligned} X = \{&Axle_F, Axle_B, Wheel_{RF}, Wheel_{LF}, Wheel_{RB}, Wheel_{LB}, Nuts_{RF}, \\ &Nuts_{LF}, Nuts_{RB}, Nuts_{LB}, Cap_{RF}, Cap_{LF}, Cap_{RB}, Cap_{LB}, Inspect\} \, . \end{aligned}$$

PRECEDENCE CONSTRAINTS

The value of each variable is the time that the task starts. Next we represent **precedence constraints** between individual tasks. Whenever a task T_1 must occur before task T_2, and task T_1 takes duration d_1 to complete, we add an arithmetic constraint of the form

$$T_1 + d_1 \leq T_2 \, .$$