CONSTRAINT SATISFACTION PROBLEMS

Chapter 6

Outline

- Constraint Satisfaction Problems (CSP)
- Backtracking search for CSPs
- Local search for CSPs

Constraint satisfaction problems (CSPs)

- **In standard search problems, states:**
  - Atomic ("black box" with no internal structure)
  - Evaluated by domain-specific heuristics
  - Tested to see whether they are goal states

- **In CSPs:** a **factored representation** for each state
  - State is defined by variables X_i with values from domain D_i
  - Goal test is a set of constraints specifying **allowable combinations of values** for subsets of variables
  - Solution: one value for each variable that satisfies all the constraints

Constraint satisfaction problems (CSPs)

- Allows useful **general-purpose algorithms** to solve complex problems
  - **with more power** than standard search algorithms
  - **general-purpose heuristics** rather than *problem-specific* heuristics

- **The main idea** of algorithms for solving **CSPs**
  - To **eliminate large portions** of the **search space** all at once
  - by identifying **variable/value combinations that violate** the constraints

Constraint satisfaction problem

- **Set of variables** $X = \{X_1, X_2, \ldots, X_n\}$

- **Set of domains** $D = \{D_1, D_2, \ldots, D_n\}$
  - Each domain **Di** consists of a set of **allowable values** for variable X_i.
  - In many cases the **domain** is assumed to be **the same** for all variables

- **Set of constraints** $C = \{\ c_i=(scope_i, rel_i) \mid i=1,\ldots,h\}$
  - **$scope_i$: subset of X**, the **variables** that are **constrained** by c_i
  - **rel_i**: is a **relation** and tells us which **simultaneous assignments of values** to variables in **$scope_i$** are **allowed**

Constraint satisfaction problem

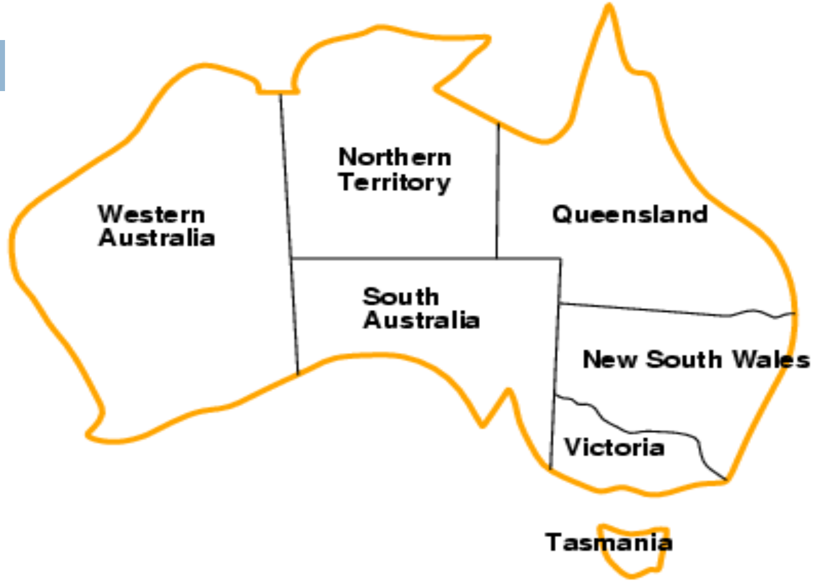
- **State:** defined by an **assignment** of values **to some or all** of the **variables**, $\{X_i = v_i, X_j = v_j, \ldots\}$

- **Assignment** can be:
  - **Consistent:** it does **not violate** any constraints
  - **Complete:** **every variable** is assigned
  - **Partial:** **only some of the variables** are assigned

- **Solution:** a **consistent and complete** assignment

Example: Map-Coloring

Coloring each region either red, green, or blue in such a way that no neighboring regions have the same color.

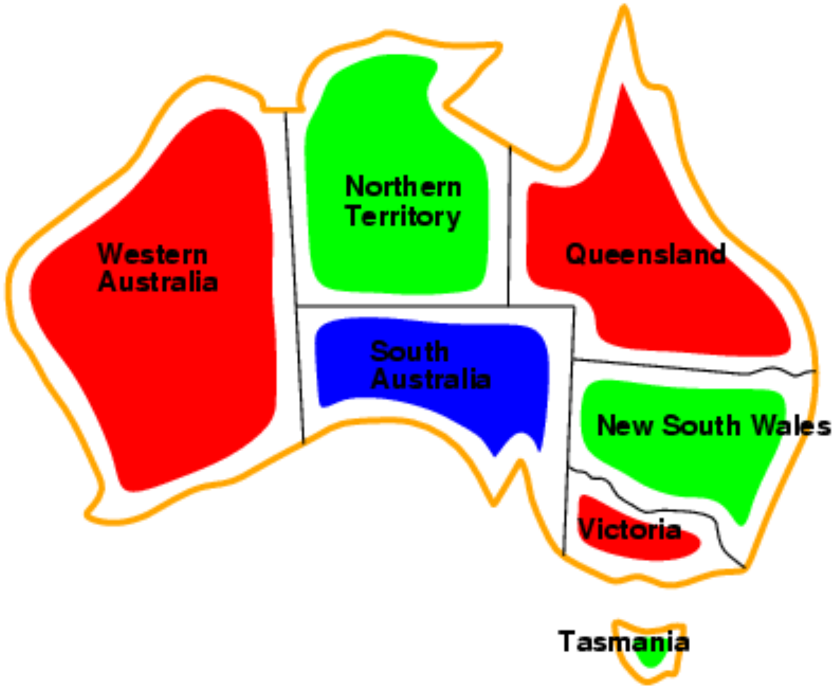


CSP formulation

- **Set of variables** $X = \{WA, NT, Q, NSW, V, SA, T\}$
- **Domain of each variabile** $D_i = \{\text{red, green, blue}\}$
- **Constraints**: **adjacent** regions must have **different colors**
  $C = \{SA \neq WA, SA \neq NT, SA \neq Q, SA \neq NSW, SA \neq V,$
  $WA \neq NT, NT \neq Q, Q \neq NSW, NSW \neq V\}$

  $WA \neq NT$, or
  (WA,NT) in
  {(red,green),(red,blue),(green,red),(green,blue),(blue,red),(blue,green)}

Example: Map-Coloring

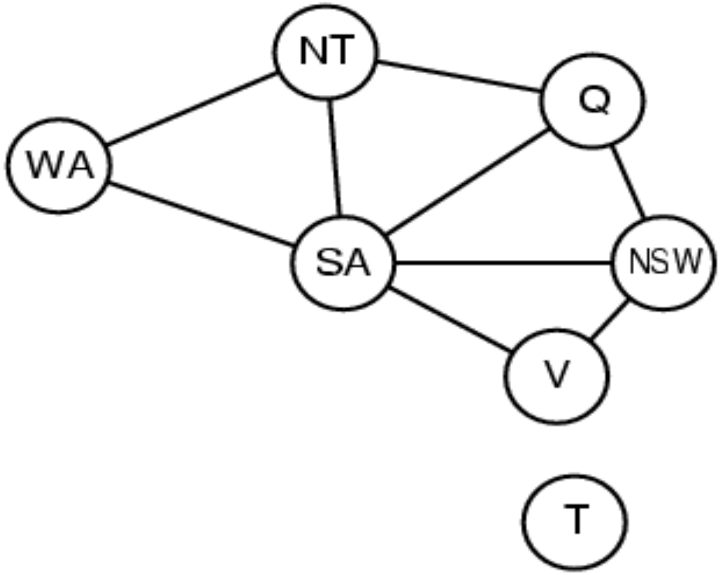


- **Solutions** are complete and consistent assignments

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

Constraint graph

- Constraint graph
  - **nodes** are variables
  - **arcs** show constraints



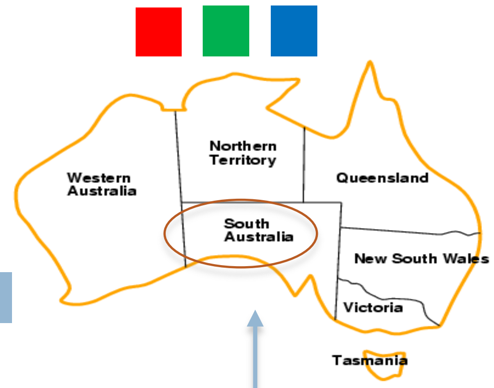
Why formulate a problem as a CSP?

- A **natural representation** for many problems

- We 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

- **CSP solvers** are **faster** than **state-space searchers** because the CSP solver can quickly eliminate large parts of the search space

Why formulate a problem as a CSP?



CSP solvers are **faster** than **state-space searchers**

Eg., if we choose {SA = blue} in the Australia problem

- None of the five neighboring variables can take blue value
- **Without constraint propagation**
  - a **search procedure** should 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

Varieties of CSPs

The **simplest** kind of **CSP** involves **variables with discrete finite domains**

- **Discrete variables**
  - **Finite domains:**
    - *n* variables, domain size *d*
    - e.g., variables *WA, NT, Q, NSW, V, SA, T* in the map coloring problem and each variable has the domain Di = {red, blue, green}
  - **Infinite domains:**
    - integers, strings, etc.
    - e.g., job scheduling, variables are start/end days for each job
    - constraints: $StartJob_1 + 5 \leq StartJob_3$
- **Continuous variables**
  - common in the real world problems, studied in the field of operations research

Varieties of constraints

- **Unary** constraints involve <u>a single variable</u>
  - e.g., SA ≠ green

- **Binary** constraints involve <u>pairs of variables</u>
  - e.g., SA ≠ WA

- **Higher-order** constraints involve <u>3 or more variables</u>

- **Global** constraints involve an <u>arbitrary number of variables</u>
  - e.g., **Alldiff**, which says that **all of the variables** involved in the constraint must have **different values**

Conversion to binary

- **Binary CSP:** CSP where each constraint relates **two variables**
- **Any CSP** can be **converted into** a CSP with only binary constraints