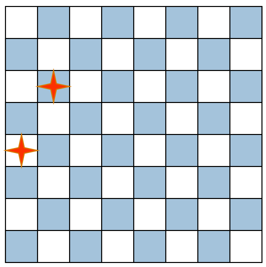
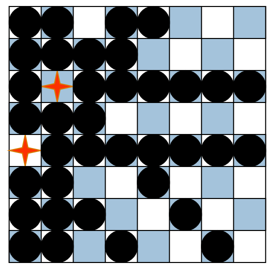
CONSTRAINT SATISFACTION

Today

- Constraint satisfaction problems (CSPs)
- Types of CSPs
- Inference
- Search

8-queens problem





Exploit the constraints

Constraint Satisfaction Problems

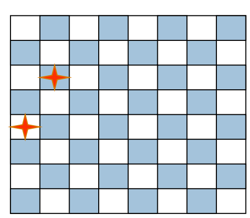
- Advantages of CSPs
  - Use general-purpose heuristics rather than problem-specific information
  - Use constraints to eliminate parts of search space
  - Detect failure early
  - We know why we failed

CSPs Defined

Example: 8-queens problems

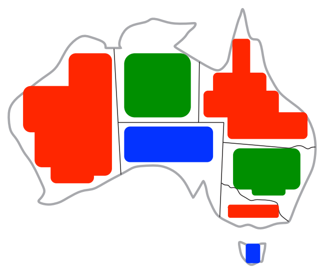
- Variables: one for each queen $\{X_1, \ldots, X_8\}$
- Domain: indicates row D = {1,2,…,8}
- Constraints:

$$X_i = k \Longrightarrow X_j \neq k \quad \forall i \neq j$$
$$X_i = k_i, X_j = k_j \Longrightarrow |i - j| \neq |k_i - k_j|$$



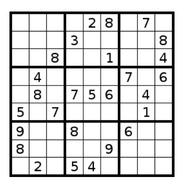
Example: Map coloring

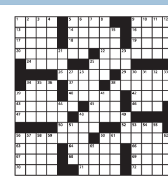
- Variables: {WA, NT, SA, Q, NSW, V, T}
- Domains: {red, blue, green}
- Constraints: adjacent regions have different colors
  - Implicit: WA ≠ NT, WA ≠ SA, SA ≠ NT, NT ≠ Q,…
  - Explicit: (WA,NT) ∈ {(red,green), (red,blue),…}



More examples

- Puzzles
  - sudoku, cryptarithmetic





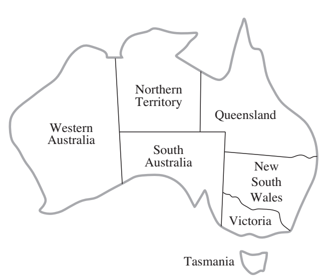
- Real-world applications
  - Assignment problems, e.g. who teaches what class?
  - Timetable problems, e.g. which class offered when? where?
  - Circuit layout
  - Workout routine app (final project)
  - Assigning first-years to rooms (final project)

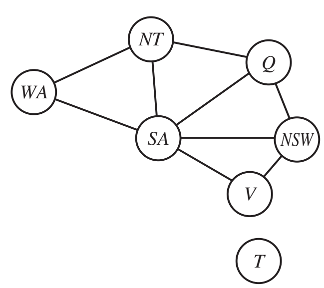
Types of constraints

- Unary constraints involve a single variable
  - e.g. SA ≠ green
- Binary constraints involve pairs of variables
  - SA ≠ NSW
  - A binary CSP can be illustrated using a constraint graph
- Higher-order constraints
  - e.g. A, B, and C cannot be in the same grouping
  - e.g. AllDiff (all variables must be assigned different values)
- Preference constraints
  - costs on individual variable assignments

Constraint Graph

- Useful for binary constraint CSPs where each constraint relates (at most) two variables
- Nodes correspond to variables
- Edges (arcs) link two variables that participate in a constraint
- Use graph to speed up search



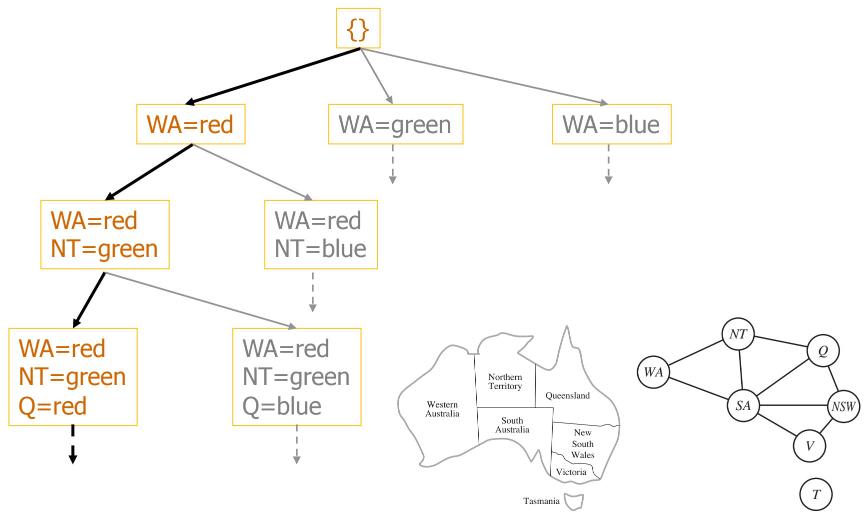


Approach 1: Inference

- Use the constraints to reduce the number of legal values for a variable
- Possibly find a solution without searching
  - Node consistency
    - A node is node-consistent if all values in its domain satisfy the unary constraints
  - Arc consistency
    - A node X_i is arc-consistent w.r.t. node X_j if for every value in D_i there exists a value in D_j that satisfies the binary constraint
    - Algorithm AC-3
  - Other types of consistency (path consistency, k-consistency, global constraints)

AC-3 algorithm for Arc consistency

Backtracking Search



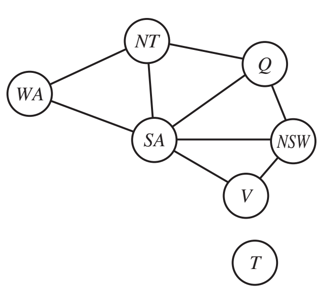
Backtracking Search Algorithm

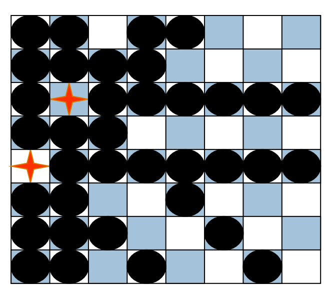
Improving Backtracking search

- Idea 1: Intelligent ordering
  - Which variable X should be assigned a value next?
  - In which order should its domain D be sorted?
- Idea 2: Incorporating inference
  - Forward checking
  - AC-3
- Idea 3: Exploiting structure
  - Can we exploit the problem structure?

Idea 1: Intelligent Ordering

- Which variable should we choose?





Idea 1: Intelligent Ordering

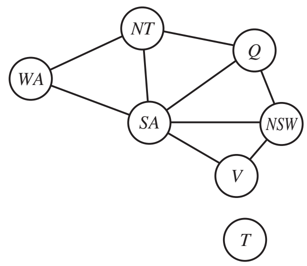
- Variable ordering
  - Minimum-remaining values heuristic - Choose the variable with the fewest "legal" moves remaining
  - Degree heuristic - Choose variable involved in the largest number of constraints with remaining unassigned variables
- Value ordering
  - Least-constraining value heuristic - Choose the value that rules out the fewest choices for the neighboring variables

Idea 2: Incorporating Inference

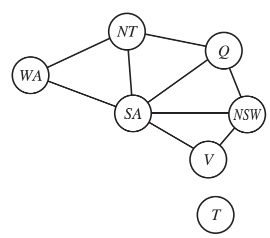
- Forward checking
  - After an assignment X = x, ensure all arcs of the form (Y,X) are arc consistent
- Run AC-3 algorithm
  - Ensure all arcs are arc consistent
- Run path-consistency or k-consistency algorithm

Example

- Run Backtracking on graph coloring
  - Use fixed ordering of variables
  - Use forward checking for inference

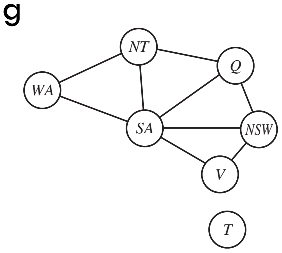


Limitations of Forward Checking



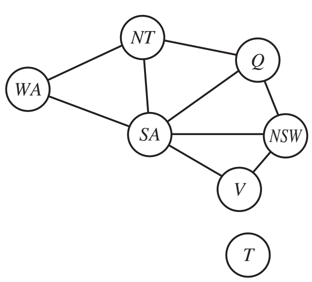
Limitations of Forward Checking

- Using AC-3 instead of forward checking



Idea 3: Exploit Structure

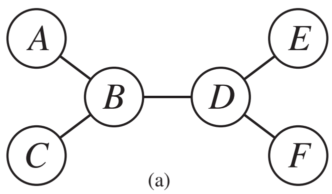
- Independent subproblems
  - Find connected components of the constraint graph
  - If we split n variables into sub-problems of c variables each then: $O(d^n) \longrightarrow O(d^c\, n/c)$

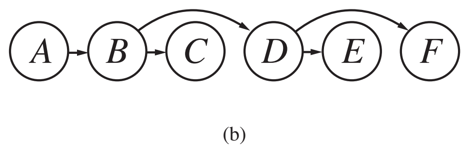


Tasmania is independent of the mainland

Idea 3: Exploit Structure

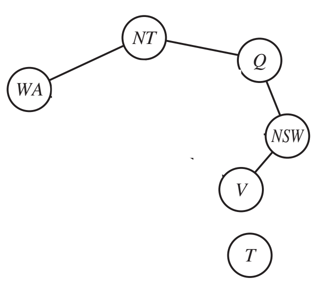
- Tree structured constraint graphs
  - Can solve in linear time using AC-3

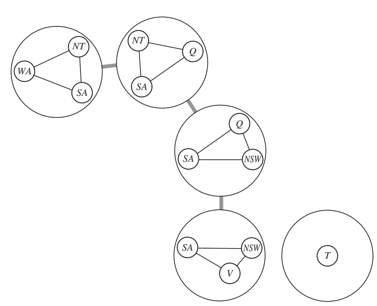




Idea 3: Exploit Structure

- Reduction to a tree structured graph
  - Cycle cutset – a subset of the variables whose removal creates a tree.
  - Tree decomposition – Divide graph into subproblems, solve independently merge the solutions





CSP Summary

- Constraint Satisfaction Problems (CSPs)
- Solving CSPs using inference
- Solving CSPs using search
  - Backtracking algorithm = DSF + fixed ordering + constraints checking
  - General (not problem-specific) heuristics
- Improving Backtracking
  - Intelligent ordering
  - Incorporating inference
  - Exploiting structure