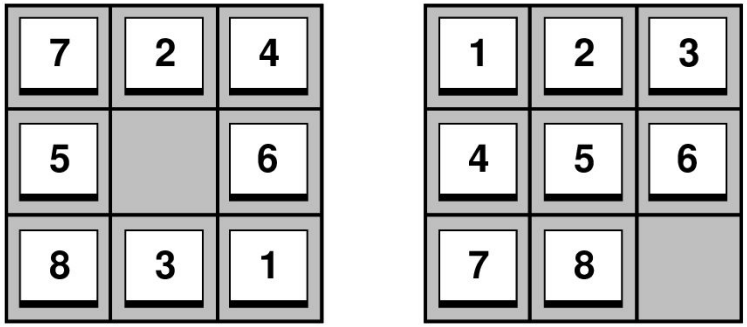
Artificial Intelligence

Admissible heuristics

E.g., for the 8-puzzle:

$h_1(n)$ = number of misplaced tiles
$h_2(n)$ = total Manhattan distance
(i.e., no. of squares from desired location of each tile)

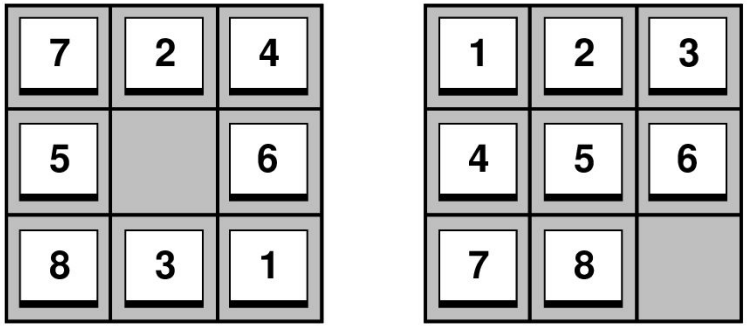


$h_1(S) =$?? 6
$h_2(S) =$?? 4+0+3+3+1+0+2+1 = 14

Admissible heuristics

E.g., for the 8-puzzle:

$h_1(n)$ = number of misplaced tiles
$h_2(n)$ = total Manhattan distance
(i.e., no. of squares from desired location of each tile)



$h_1(S) =$?? 6
$h_2(S) =$?? 4+0+3+3+1+0+2+1 = 14

If $h_2(n) \geq h_1(n)$ for all n (both admissible)
then h_2 dominates h_1 and is better for search

Properties of greedy search

Complete?? No–can get stuck in loops, e.g.,

Complete in finite space with repeated-state checking

Time?? $O(b^m)$, but a good heuristic can give dramatic improvement

Space?? $O(b^m)$—keeps all nodes in memory

Optimal?? No

Properties of A*

Complete?? Yes, unless there are infinitely many nodes with $f \leq f(G)$

Time?? Exponential in [relative error in h $\times$ length of soln.]

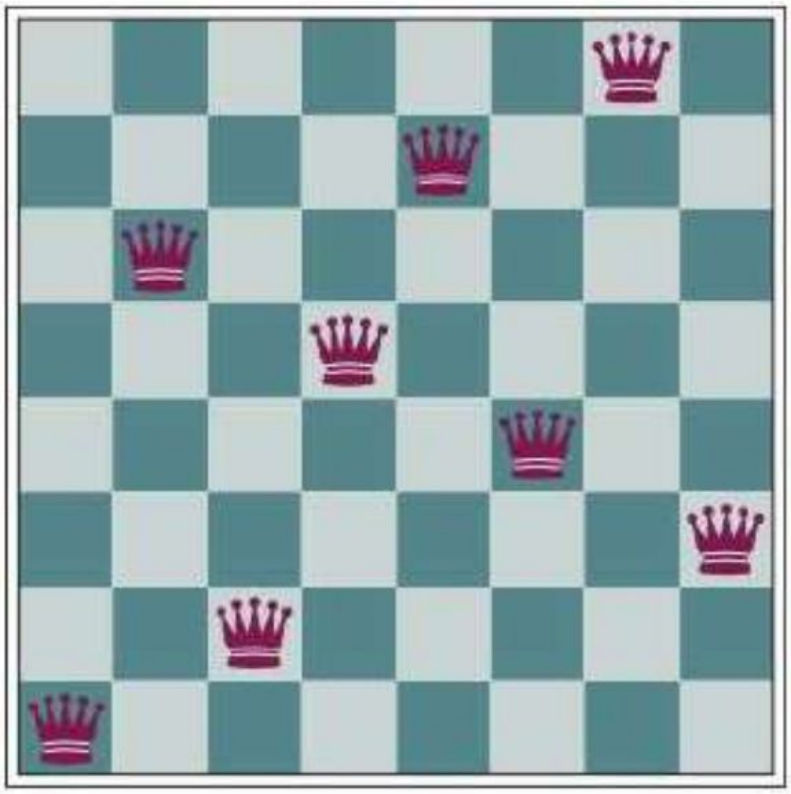
Space?? O(b^m), as in Greedy Best-First — may end up with all nodes in memory

Optimal?? Yes—cannot expand f_{i+1} until f_i is finished

If h is admissible

Hill climbing Search (greedy local search)

Hill climbing is sometimes called greedy local search because it grabs a good neighbor state without thinking ahead.



Just five steps

Local search

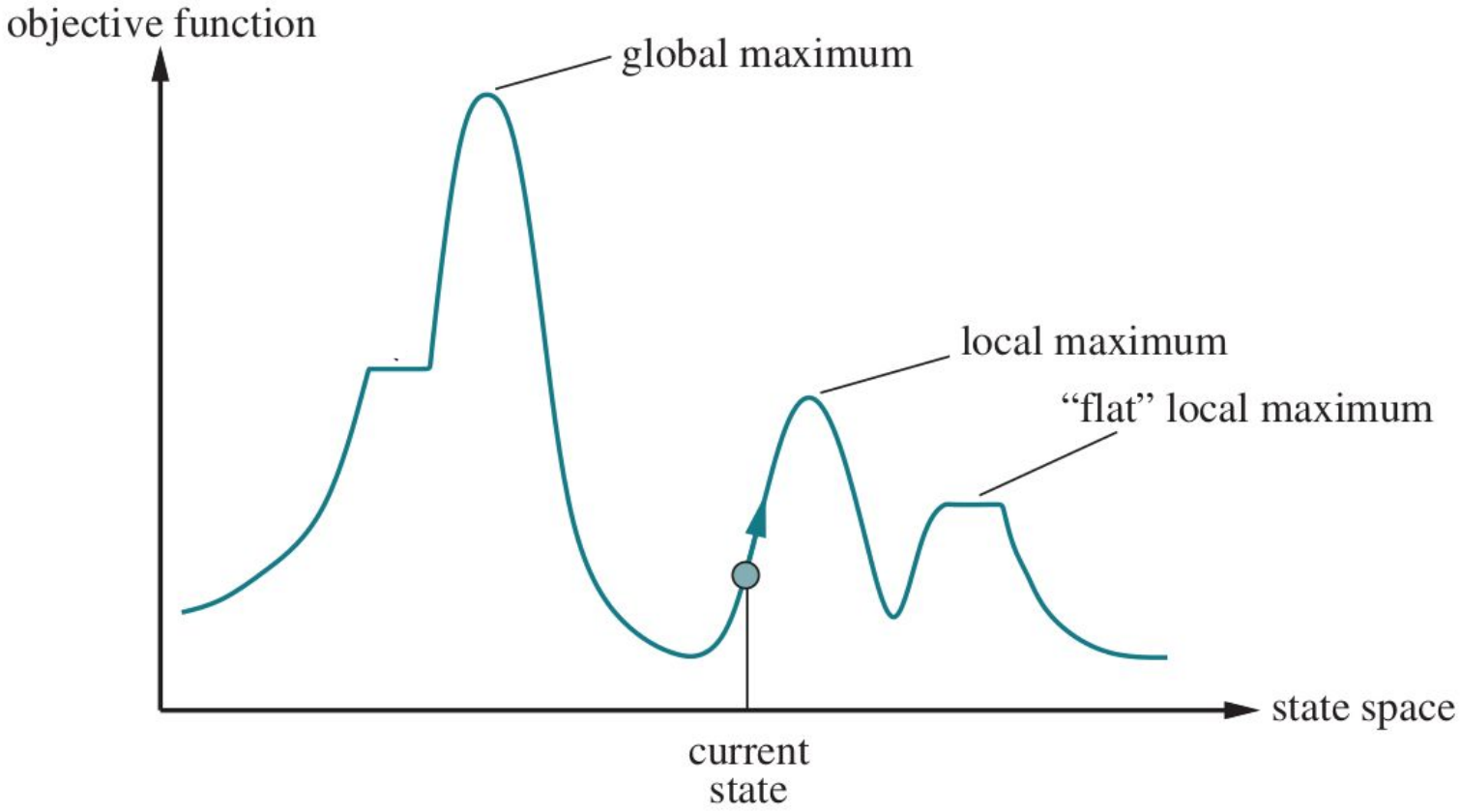
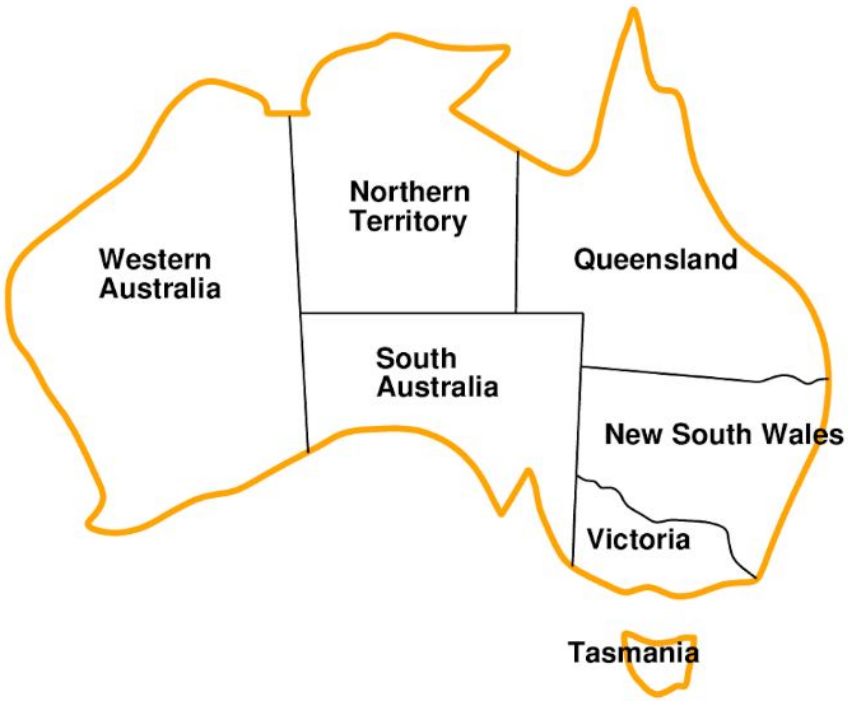


Figure 4.1 A one-dimensional state-space landscape in which elevation corresponds to the objective function. The aim is to find the global maximum.

Constraint Satisfaction Problem

Constraint satisfaction problems (CSPs)

- CSPs are mathematical questions defined as a set of objects whose state must satisfy a number of constraints.



Constraint satisfaction problems (CSPs)

- CSPs are mathematical questions defined as a set of objects whose state must satisfy a number of constraints.
- Standard search problem considers the states, actions, successor function, and goal test.

Constraint satisfaction problems (CSPs)

- CSPs are mathematical questions defined as a set of objects whose state must satisfy a number of constraints.
- Standard search problem considers the states, actions, successor function, and goal test.
- In CSP, state is defined by the variables with values from domain.
- Goal test relates to satisfying constraints that conditions allowable combination of values for the subset of variables.

Constraint satisfaction problems (CSPs)

Constraint satisfaction problem (CSP) consists of following three components.

$\mathcal{X}$ is a set of variables, $\{X_1, \ldots, X_n\}$.

$\mathcal{D}$ is a set of domains, $\{D_1, \ldots, D_n\}$, one for each variable.

$\mathcal{C}$ is a set of constraints that specify allowable combinations of values.

Constraint satisfaction problems (CSPs)

Constraint satisfaction problem (CSP) consists of following three components.

$\mathcal{X}$ is a set of variables, $\{X_1, \ldots, X_n\}$.

$\mathcal{D}$ is a set of domains, $\{D_1, \ldots, D_n\}$, one for each variable.

$\mathcal{C}$ is a set of constraints that specify allowable combinations of values.

The domain D_i consists of a set of allowable values $\{v_1 \ldots, v_k\}$ for variable X_i.

Constraint satisfaction problems (CSPs)

Constraint satisfaction problem (CSP) consists of following three components.

$\mathcal{X}$ is a set of variables, $\{X_1, \ldots, X_n\}$.

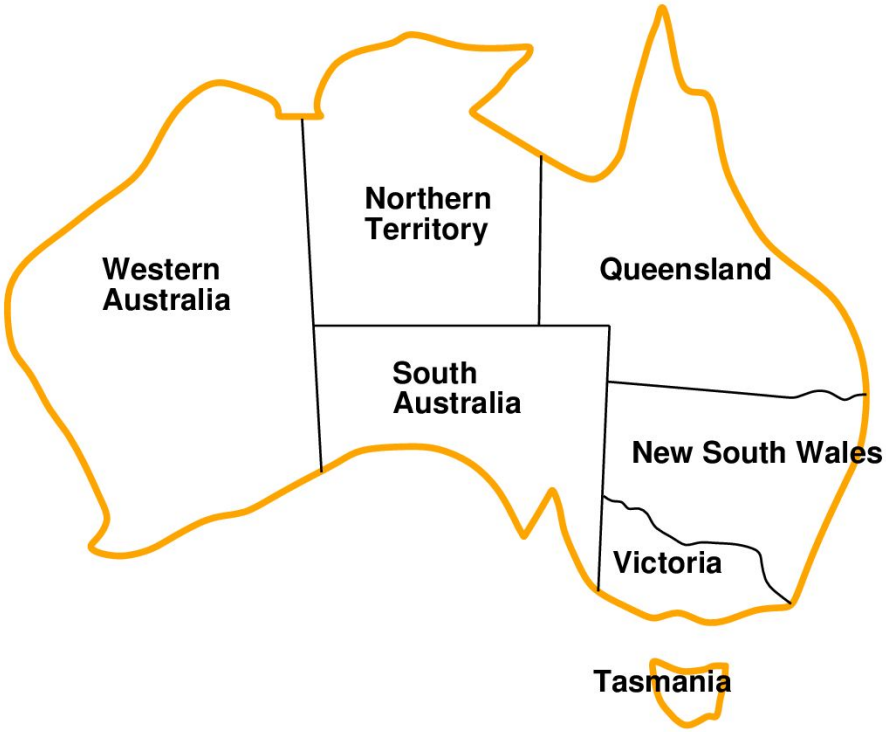
$\mathcal{D}$ is a set of domains, $\{D_1, \ldots, D_n\}$, one for each variable.

$\mathcal{C}$ is a set of constraints that specify allowable combinations of values.

The domain D_i consists of a set of allowable values $\{v_1 \ldots, v_k\}$ for variable X_i.

CSPs deal with assignments of values to variables, $\{X_i = v_i, X_j = v_j, \ldots\}$. while satisfying the constraints.

Example: Map coloring

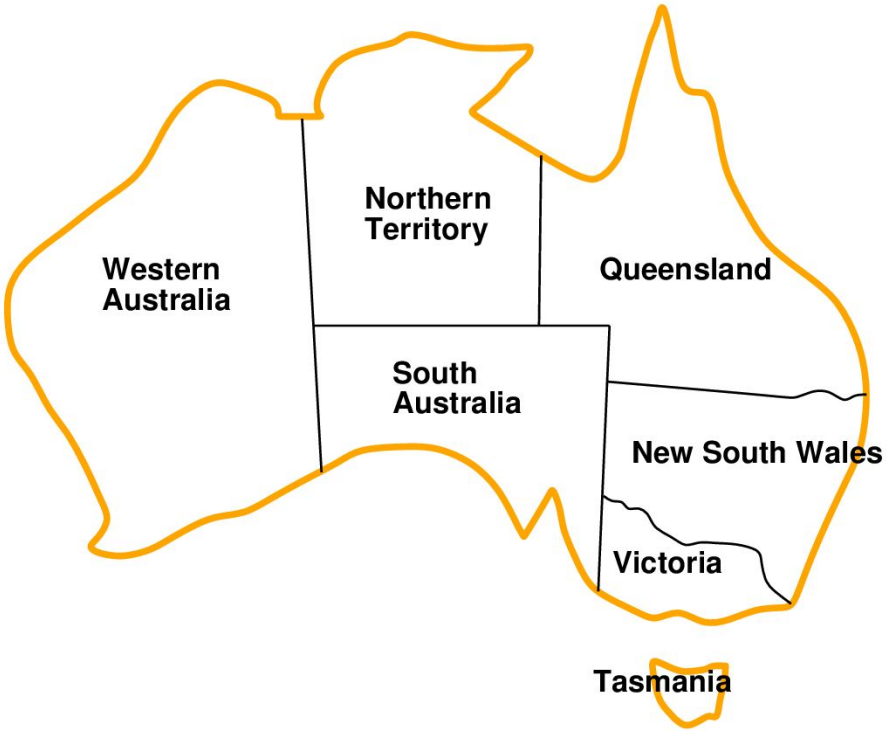


Variables:

Domains:

Constraints:

Example: Map coloring



Variables: MA, NT, Q, NSW, V, SA, T

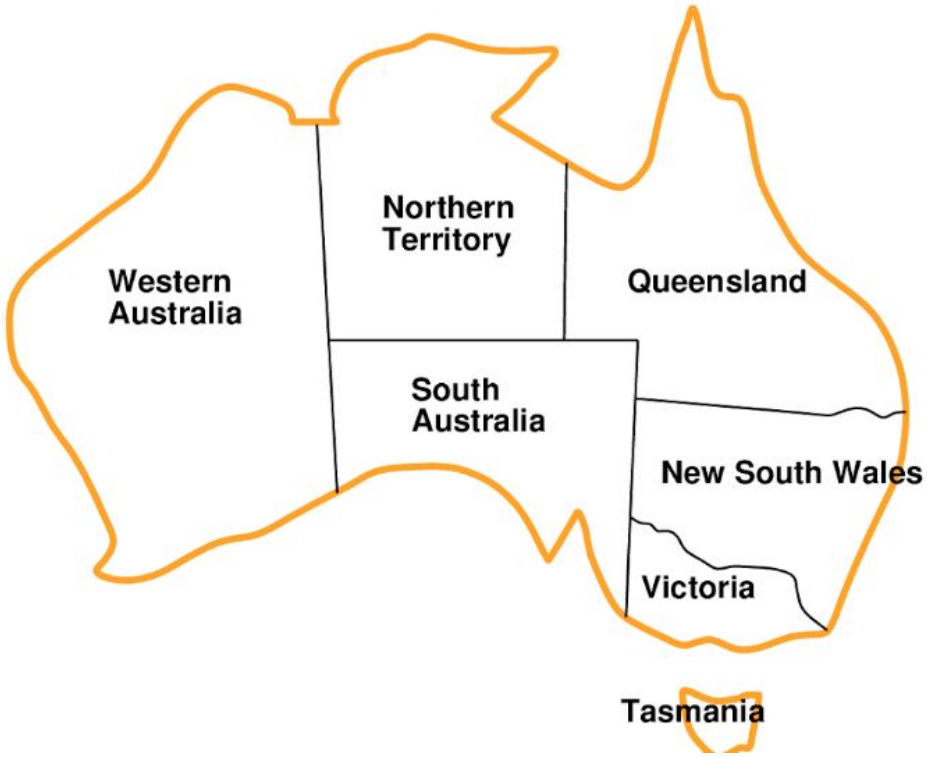
Domains: $D_i = \{r(ed), g(reen), b(lue)\}$

Constraints: adjacent regions must have different colors

e.g., $WA \neq NT$ (if the language allows this), or

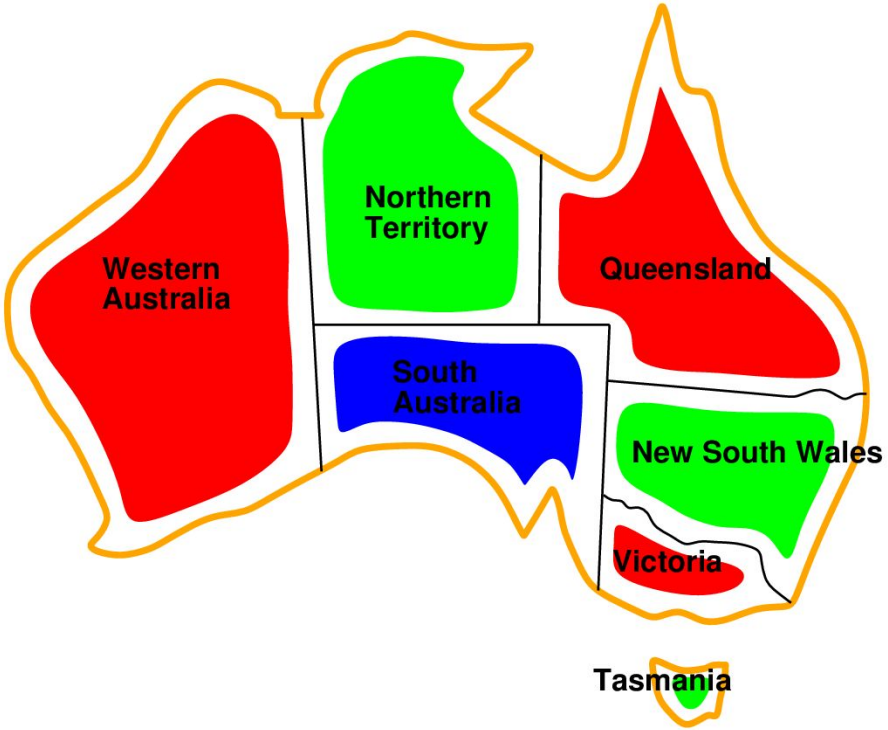
$(WA, NT) \in \{(r, g), (r, b), (g, r), (g, b), \ldots\}$

Example: Map coloring contd.



Solutions are assignments satisfying all constraints,

Example: Map coloring contd.



Solutions are assignments satisfying all constraints,

e.g., $\{WA = r, NT = g, Q = r, NSW = g, V = r, SA = b, T = g\}$

Varieties of constraints

Unary constraints involve a single variable

e.g., $SA \neq g$

Binary constraints involve pairs of variables

e.g., $SA \neq WA$

Varieties of constraints

Unary constraints involve a single variable

e.g., $SA \neq g$

Binary constraints involve pairs of variables

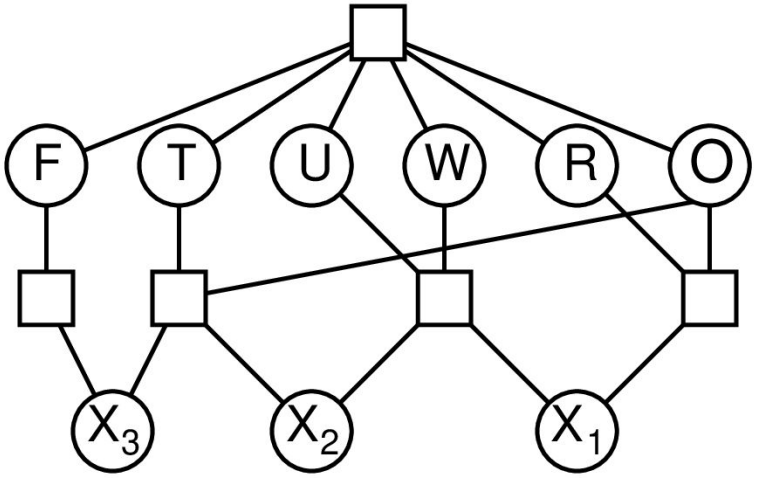
e.g., $SA \neq WA$

Higher-order constraints involve 3 or more variables

e.g., cryptarithmetic column constraints

Example: Cryptarithmetic

$$\begin{array}{r} \ T\ W\ O \\ +\ T\ W\ O \\ \hline F\ O\ U\ R \end{array}$$



Variables: $F,\ T,\ U,\ W,\ R,\ O,\ X_1,\ X_2,\ X_3$

Domain: $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Constraints: $alldiff\,(F, T, U, W, R, O)$

$O + O = R + 10 \cdot X_1$

$\ldots$

Varieties of constraints

Unary constraints involve a single variable

e.g., $SA \neq g$

Binary constraints involve pairs of variables

e.g., $SA \neq WA$

Higher-order constraints involve 3 or more variables

e.g., cryptarithmetic column constraints

Preferences are soft constraints

e.g., red is better than $green$

often representable by a cost for each variable assignment

$\rightarrow$ constrained optimization problems

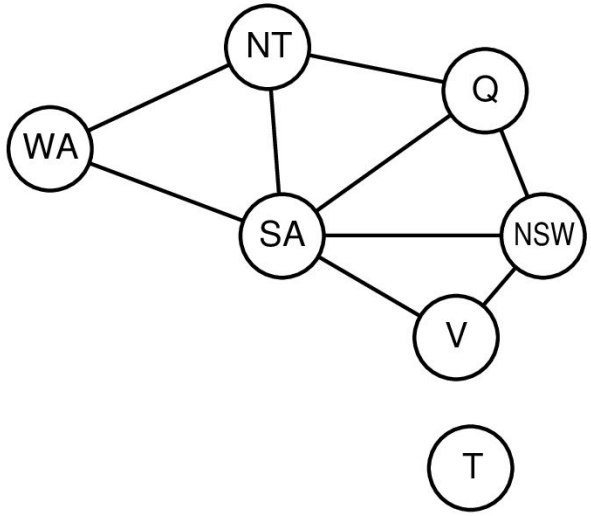
Constraint Satisfaction

- 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 value, and a solution to a CSP is a consistent, **complete assignment**.
- A partial assignment is one that leaves some variables **unassigned**, and a **partial solution** is a partial assignment that is consistent.

Constraint graph

Binary CSP: each constraint relates at most two variables

Constraint graph: nodes are variables, arcs show constraints

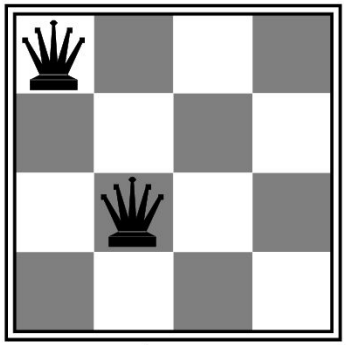


General-purpose CSP methods use the graph structure to speed up search

e.g., Tasmania is an independent subproblem!

Example: 4-Queens as a CSP

Assume one queen in each column. Which row does each one go in?



$Q_1 = 1 \quad Q_2 = 3$

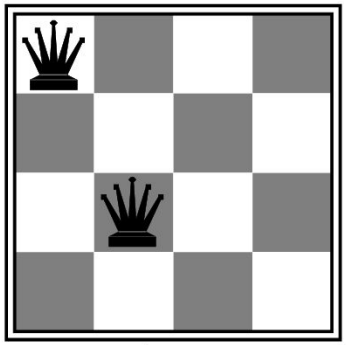
Variables

Domains

Constraints

Example: 4-Queens as a CSP

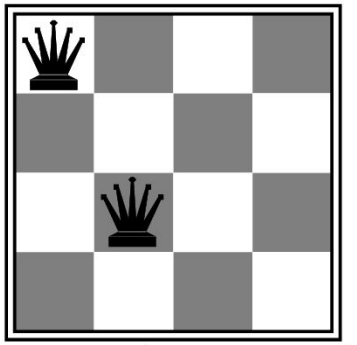
Assume one queen in each column. Which row does each one go in?



Variables	Q_1, Q_2, Q_3, Q_4	
Domains	$D_i = \{1, 2, 3, 4\}$	
Constraints	$Q_i \neq Q_j$	(cannot be in same row)
	$\lvert Q_i - Q_j \rvert \neq \lvert i - j \rvert$	(cannot be on same diagonal)

Example: 4-Queens as a CSP

Assume one queen in each column. Which row does each one go in?



Variables	Q_1, Q_2, Q_3, Q_4	
Domains	$D_i = \{1, 2, 3, 4\}$	
Constraints	$Q_i \neq Q_j$	(cannot be in same row)
	$\lvert Q_i - Q_j \rvert \neq \lvert i - j \rvert$	(cannot be on same diagonal)

Translate each constraint into set of allowable values for its variables
E.g., values for (Q_1, Q_2) are $(1,3)$ $(1,4)$ $(2,4)$ $(3,1)$ $(4,1)$ $(4,2)$

Standard search formulation (incremental)

Let's start with a basic, naive approach and then improve it

States are defined by the values assigned so far

Initial state: the empty assignment, $\{\}$

Successor function: assign a value to an unassigned variable that does not conflict with current assignment.
Fail if no legal assignments (not fixable!)

Goal test: the current assignment is complete

Standard search formulation (incremental)

Let's start with a basic, naive approach and then improve it

States are defined by the values assigned so far

Initial state: the empty assignment, $\{\}$

Successor function: assign a value to an unassigned variable that does not conflict with current assignment.
Fail if no legal assignments (not fixable!)

Goal test: the current assignment is complete

Note:

1. This is the same for all CSPs!
2. Every solution appears at depth n with n variables $\Longrightarrow$ use depth-first search

Backtracking search

Order of variables in variable assignments is irrelevan
i.e., $(WA = r,\ NT = g)$ same as $(NT = g,\ WA = r)$

Only need to consider assignments to a single variable at each node
$\Longrightarrow\ b = d$ and there are d^n leaves

Depth-first search for CSPs with single-variable assignments is called backtracking search

Backtracking search is the basic uninformed algorithm for CSPs

Real World CSPs

- Class Assignment problems: who teaches what class
- Scheduling problems: Which class is offered when and where
- Transportation Scheduling and trips planning
- Many more…

Variable, domain, constraints?

Backtracking Search

Use depth-first-search

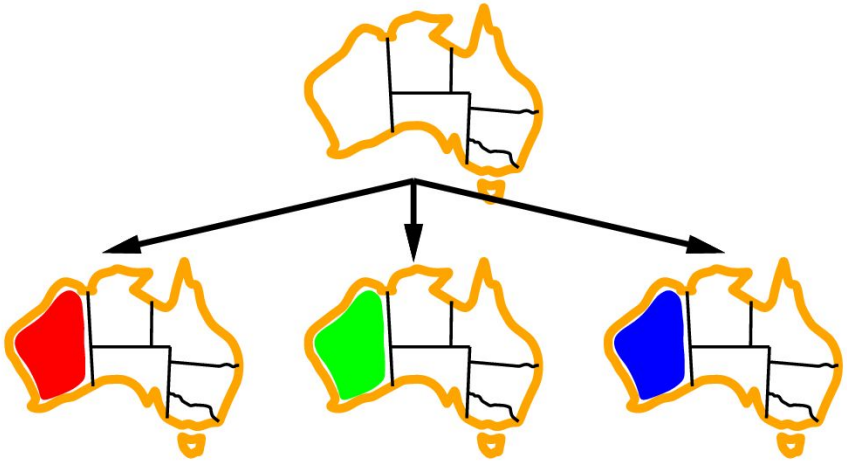
(1) Fix the **order** of assignments

(2) Check for constraint violation and constraint satisfaction when expanding the other nodes

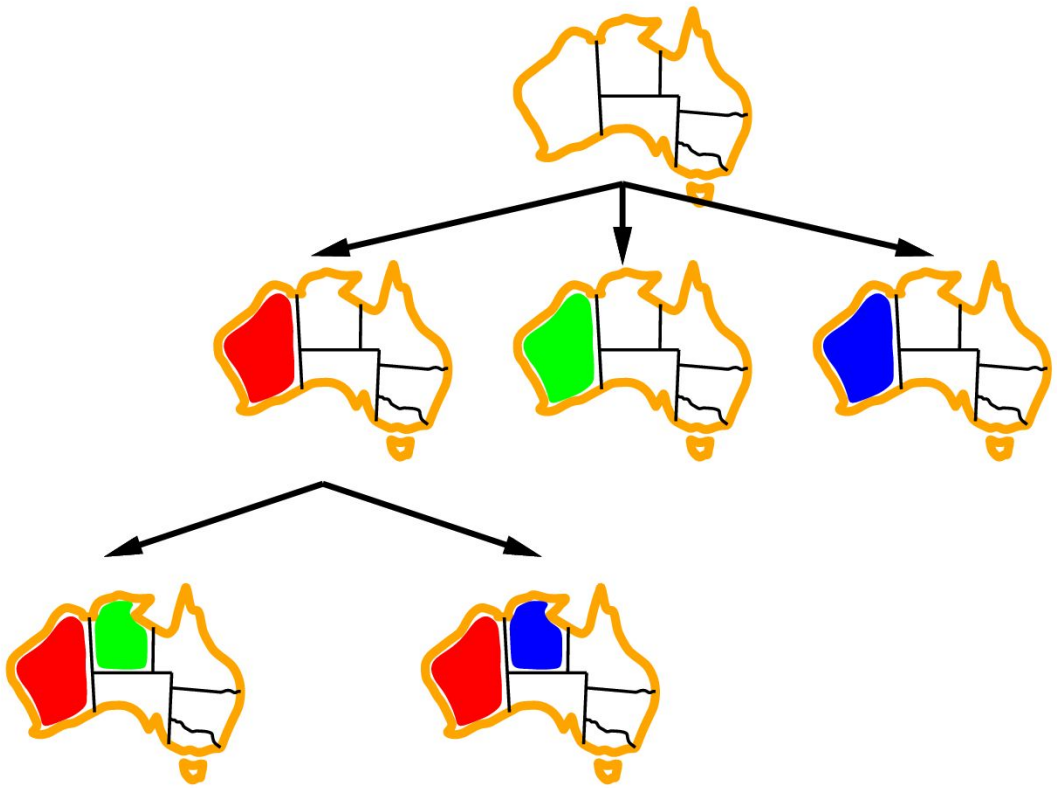
(3) Backtracking is the basic uninformed algorithm for CSPs

Backtracking example

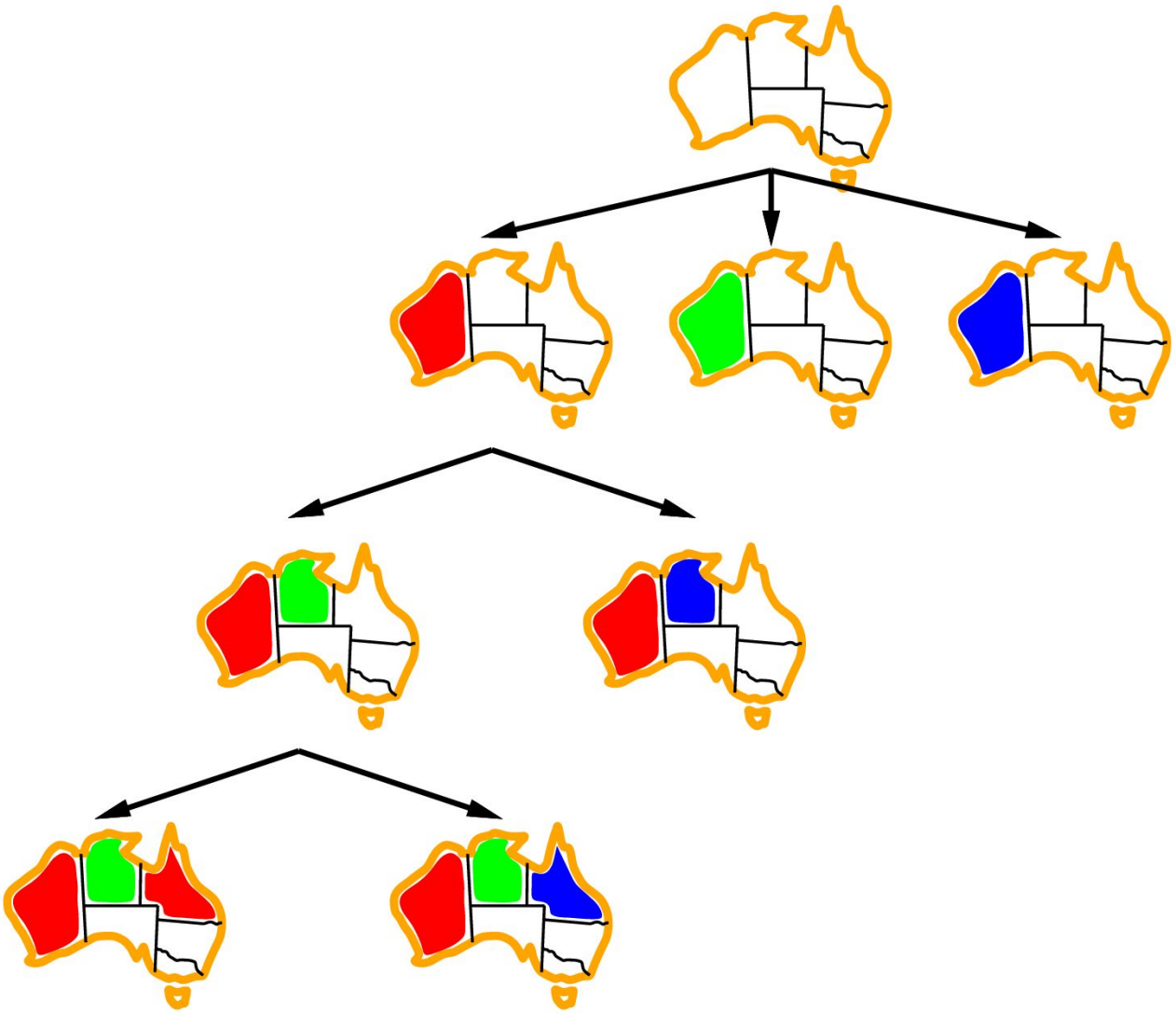
Backtracking example



Backtracking example



Backtracking example



Variable choice heuristics

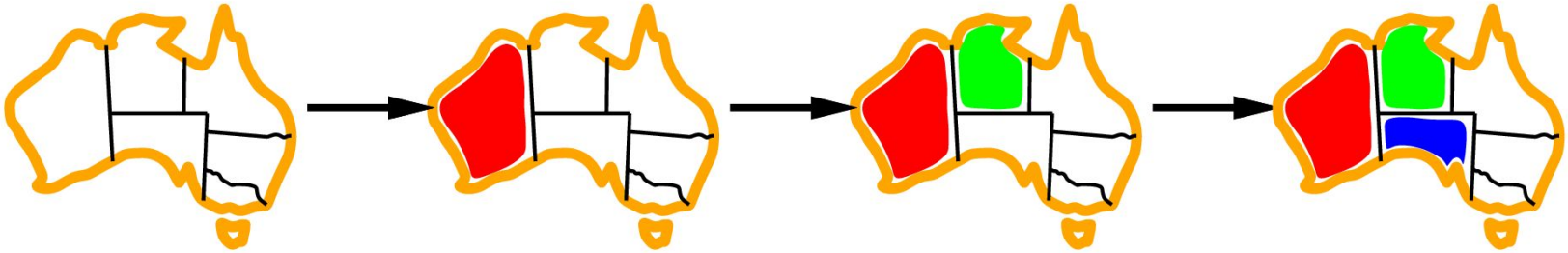
Minimum remaining values (MRV):

choose the variable with the fewest legal values

Variable choice heuristics

Minimum remaining values (MRV):

choose the variable with the fewest legal values



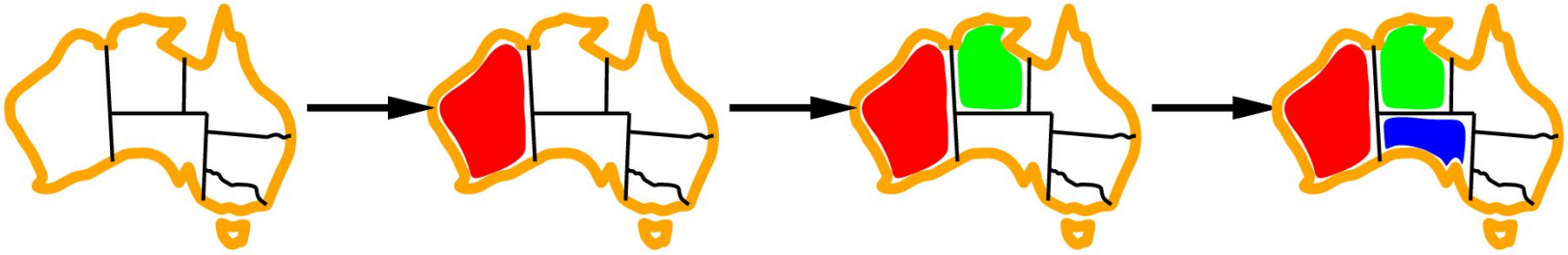
Degree heuristic:

choose the variable with the most constraints on remaining vars

Variable choice heuristics

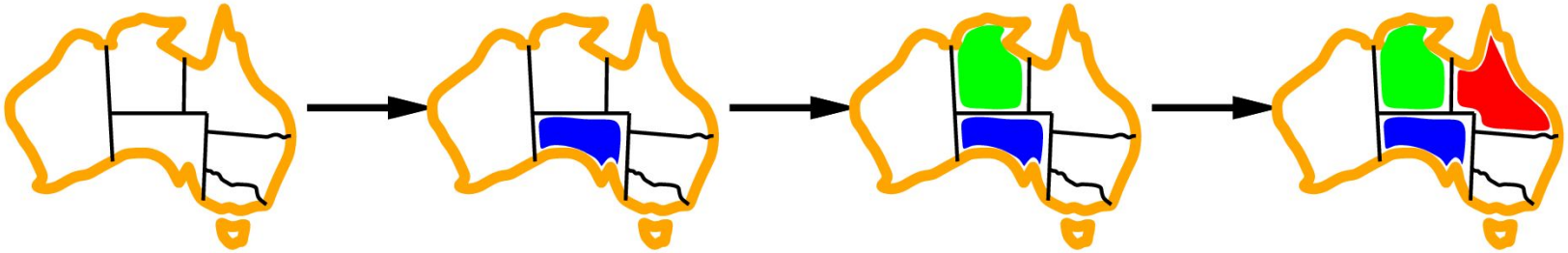
Minimum remaining values (MRV):

choose the variable with the fewest legal values



Degree heuristic:

choose the variable with the most constraints on remaining vars



Value choice heuristics

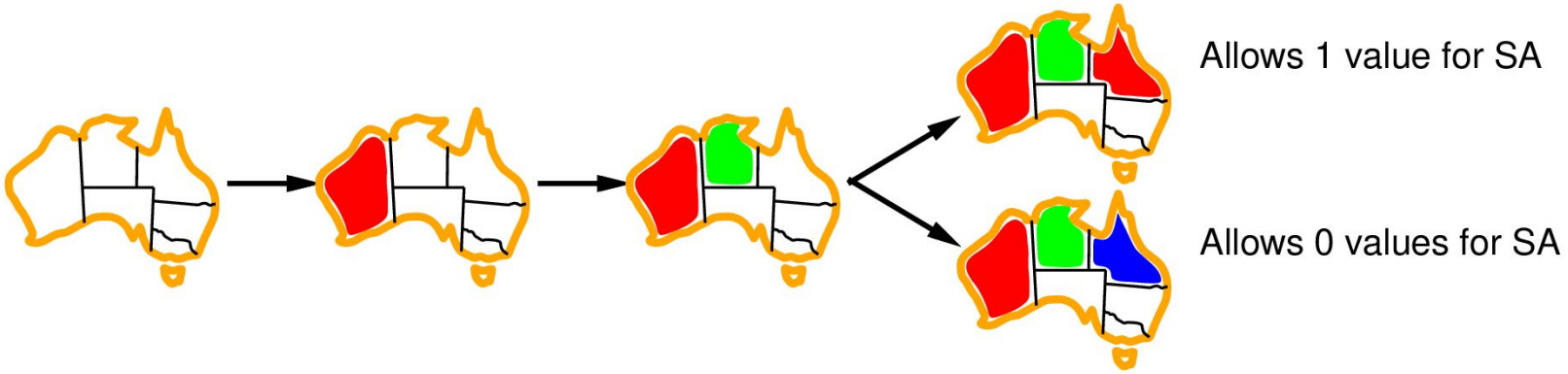
Least constraining value:

For a given a variable, choose the least constraining value: the one that rules out the fewest values in the remaining variables

Value choice heuristics

Least constraining value:

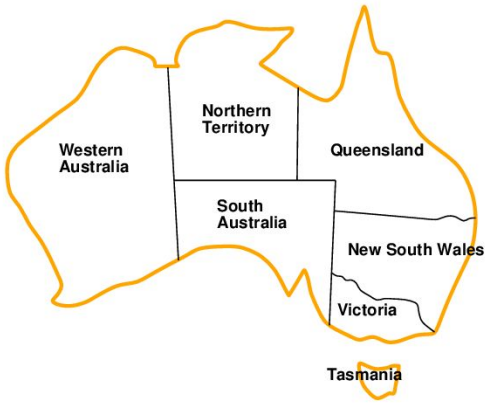
For a given a variable, choose the least constraining value: the one that rules out the fewest values in the remaining variables



Forward checking

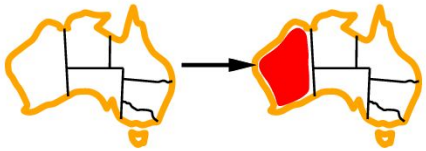
Idea: Keep track of remaining legal values for unassigned variables

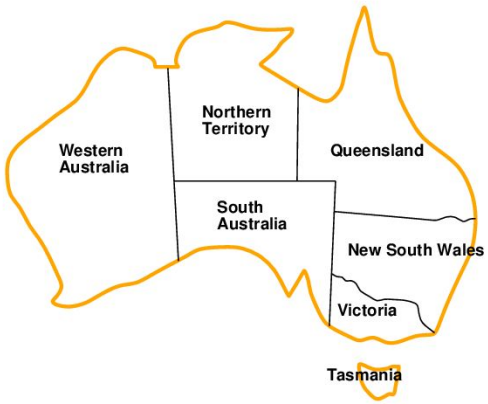
Terminate search when any variable has no legal values



Forward checking

Idea: Keep track of remaining legal values for unassigned variables
Terminate search when any variable has no legal values

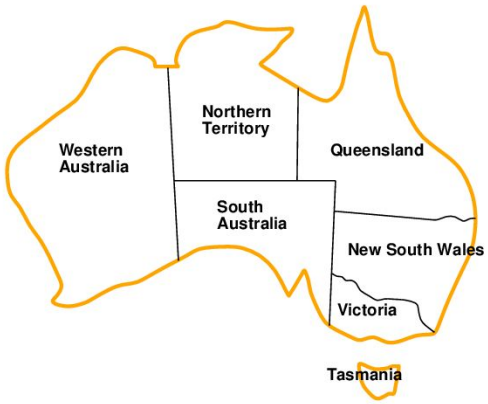




Forward checking

Idea: Keep track of remaining legal values for unassigned variables

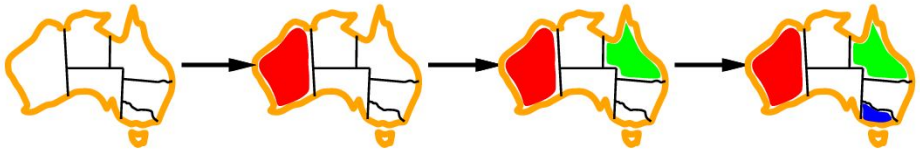
Terminate search when any variable has no legal values

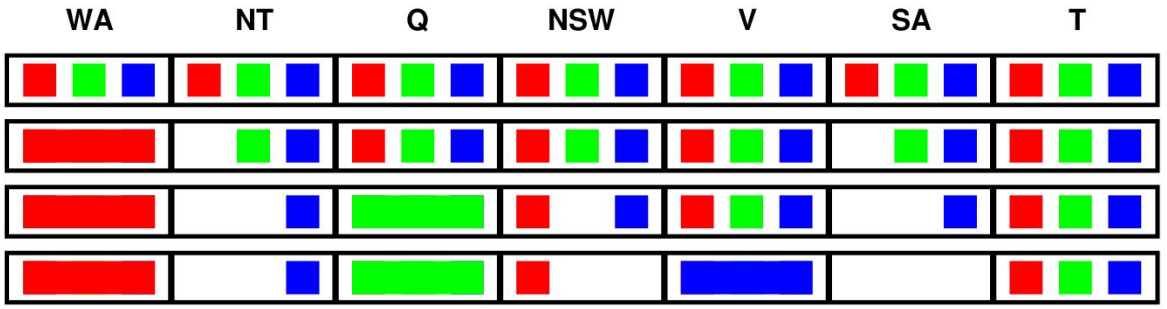


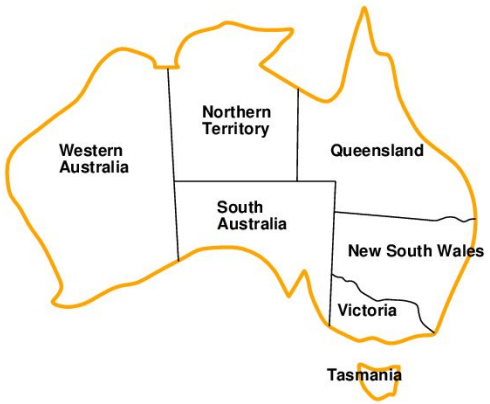
Forward checking

Idea: Keep track of remaining legal values for unassigned variables

Terminate search when any variable has no legal values

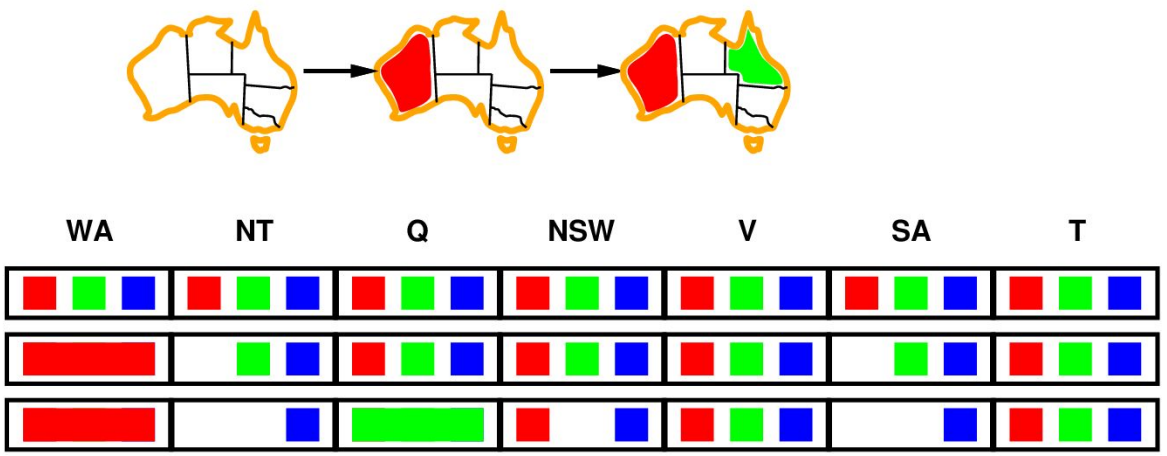






Constraint propagation

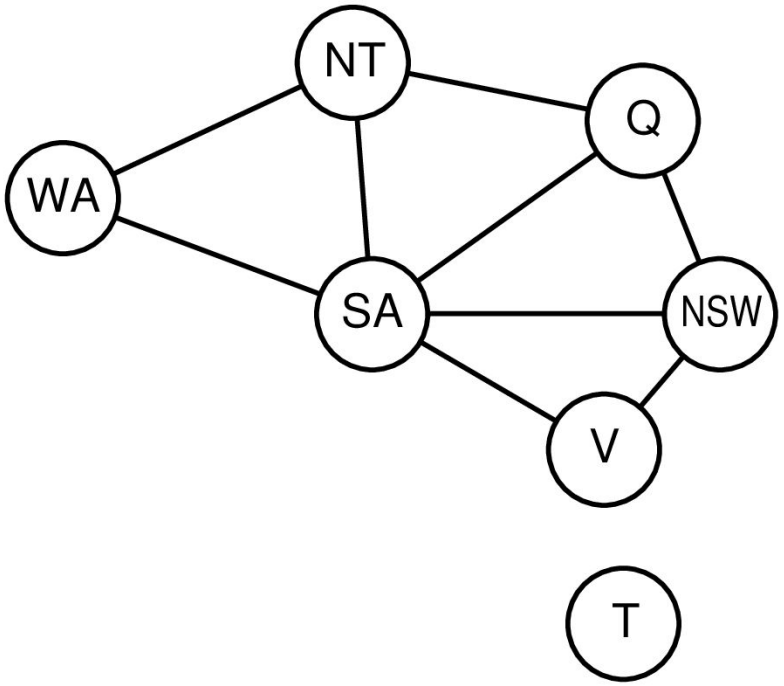
Forward checking propagates information from assigned to unassigned variables, but doesn't provide early detection for all failures:



NT and SA cannot both be blue!

Constraint propagation repeatedly enforces constraints locally

Problem structure

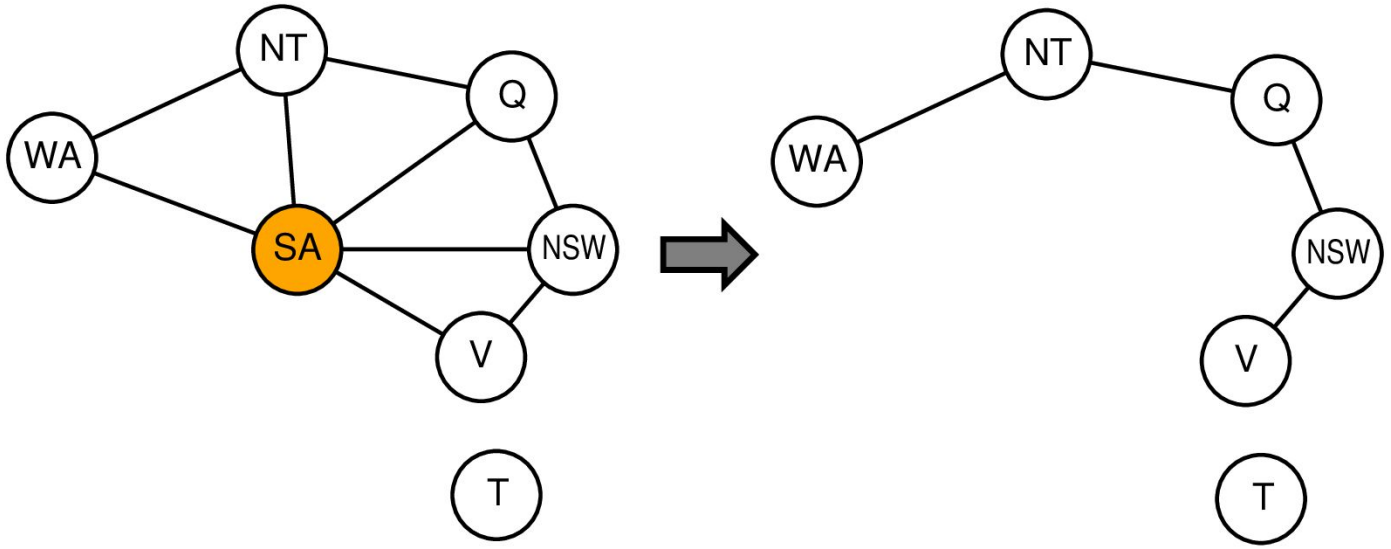


Tasmania and mainland are independent subproblems

Identifiable as connected components of constraint graph

Nearly tree-structured CSPs

Conditioning: instantiate a variable, prune its neighbors' domains



Cutset conditioning: instantiate (in all ways) a set of variables so that the remaining constraint graph is a tree

Constrained optimization problem

Constrained minimization problem

A general constrained minimization problem may be written as follows.

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_i(\mathbf{x}) = c_i \quad \text{for } i = 1, \ldots, n \quad \text{Equality constraints} \\ & h_j(\mathbf{x}) \geqq d_j \quad \text{for } j = 1, \ldots, m \quad \text{Inequality constraints} \end{aligned}$$

- Where above two constraints are constraints (hard) that required to be satisfied.
- $f(\mathbf{x})$ is the objective function that needs to be optimized subject to the constraints.

Unconstrained optimization problem $\min_x F(x)$ or $\max_x F(x)$

Constrained minimization example

$$\min f(\mathbf{x}) = x_1^2 + x_2^4$$

subject to

$$x_1 \geq 1$$

and

$$x_2 = 1,$$

where $\mathbf{x}$ denotes the vector (x_1, x_2).

Suppose, $(x_1, x_2) = (0, 0)$ and $(x_1, x_2) = (1, 1)$.

Constraint optimization

- The constrained-optimization problem (COP) is a significant generalization of the classic constraint-satisfaction problem (CSP).

Constraint optimization

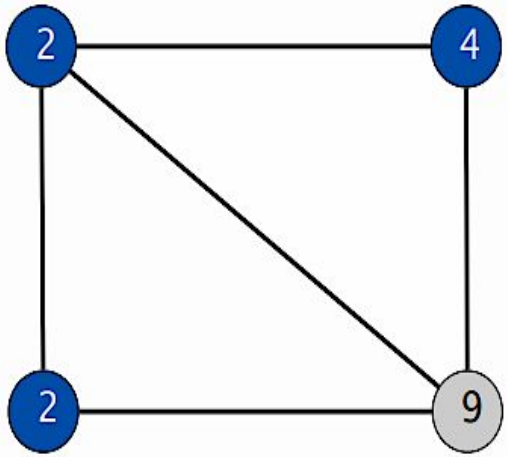
- The constrained-optimization problem (COP) is a significant generalization of the classic constraint-satisfaction problem (CSP).
- COP is a CSP that includes an objective function to be optimized.
- COP is the process of optimizing an objective function with respect to some variables in the presence of constraints on those variables.

Constraint optimization

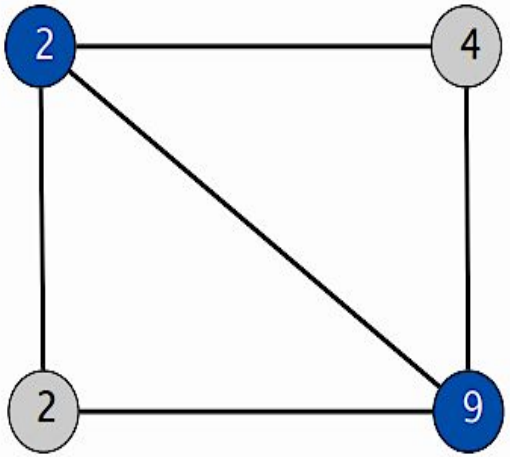
- The constrained-optimization problem (COP) is a significant generalization of the classic constraint-satisfaction problem (CSP).
- COP is a CSP that includes an objective function to be optimized.
- COP is the process of optimizing an objective function with respect to some variables in the presence of constraints on those variables.
- The objective function is either a cost function, which is to be minimized, or a reward function or utility function, which is to be maximized.
- Typical constraints might involve time, money, and resources.

Constrained minimization example

Weighted vertex cover. Given a graph G with vertex weights, find a vertex cover of minimum weight.



weight = 2 + 2 + 4



weight = 11