Lecture 7 of 42

Intro to Constraint Satisfaction Problems (CSP) & CSP Search
Discussion: Search Recap

William H. Hsu
Department of Computing and Information Sciences, KSU

KSOL course page: http://snipurl.com/v9v3
Course web site: http://www.kddresearch.org/Courses/CIS730
Instructor home page: http://www.cis.ksu.edu/~bhsu

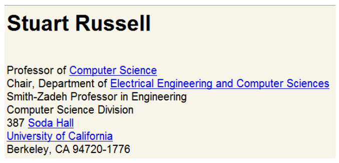
Reading for Next Class:

Sections 5.4 – 5.5, p. 151 - 158, Russell & Norvig 2nd edition
Missionaries and Cannibals: http://tr.im/yDoJ (article), http://tr.im/yDo2 (game)
Farmer, Fox, Goose, and Grain: http://tr.im/yDom (article)

Lecture Outline

- **Reading for Next Class: Sections 5.4 – 5.5, p. 151 – 158, R&N 2e**
- **Last Class: Sections 4.3**
  - **Problems in heuristic search: foothills (local optima), plateaus, ridges**
  - **Macro operators (macrops)**
    - **Encode reusable sequences of steps**
    - **Compare: hierarchical decomposition planning**
  - **Wide world of global optimization: simulated annealing, simple GA**
- **Today: Chapter 5 on Constraint Satisfaction Problems**
  - **CSPs: definition, examples**
  - **Heuristics for variable selection, value selection**
  - **Two algorithms: backtracking search, forward checking**
- **Coming Week: Chapter 4 concluded; Chapter 5**
  - **State space search: graph vs. constraint representations**
  - **Arc consistency algorithm**
  - **Search and games (start of Chapter 6)**
- **Later On: More Genetic and Evolutionary Computation (GEC)**

Acknowledgements



Stuart Russell

Professor of Computer Science
Chair, Department of Electrical Engineering and Computer Sciences
Smith-Zadeh Professor in Engineering
Computer Science Division
387 Soda Hall
University of California
Berkeley, CA 94720-1776

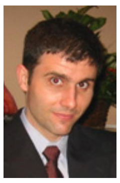
© 2004-2005

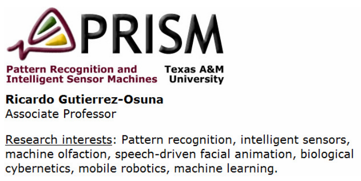
Russell, S. J.
University of California, Berkeley
http://www.eecs.berkeley.edu/~russell/

Peter Norvig
Director of Research Google

Norvig, P.
http://norvig.com/

Slides from:
http://aima.eecs.berkeley.edu





PRISM
Pattern Recognition and Intelligent Sensor Machines — Texas A&M University

Ricardo Gutierrez-Osuna
Associate Professor

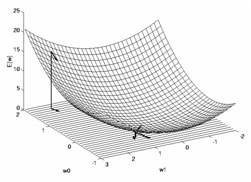
Research interests: Pattern recognition, intelligent sensors, machine olfaction, speech-driven facial animation, biological cybernetics, mobile robotics, machine learning.

© 2005 R. Gutierrez-Osuna
Texas A&M University
http://research.cs.tamu.edu/prism/

Hill-Climbing: Review

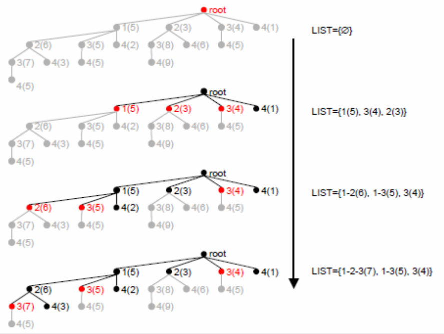
- function *Hill-Climbing* (*problem*) returns solution state
  - inputs: *problem*: specification of problem (structure or class)
  - static: *current*, *next*: search nodes
  - *current* ← *Make-Node* (*problem.Initial-State*)
  - loop do
    - *next* ← a highest-valued successor of *current*
    - if *next.value*() < *current.value*() then return *current*
    - *current* ← *next* // make transition
  - end
- Steepest Ascent Hill-Climbing
  - *aka* gradient ascent (descent)
  - Analogy: finding "tangent plane to objective surface"
  - Implementations
    - Finding derivative of (differentiable) *f* with respect to parameters
    - Example: error backpropagation in artificial neural networks (later)
- Discussion: Difference Between Hill-Climbing, Best-First?

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n}\right]$$



Beam Search: Review

- **The example below illustrates BS for a 4-dimensional search space and a queue of size 3**
  - BS cannot guarantee that the optimal subset is found:
    - In the example, the optimal is 2-3-4 (J=9), which is never explored
    - However, with the proper queue size, Beam Search can avoid getting trapped in local minimal by preserving solutions from varying regions in the search space

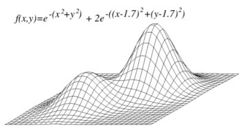


Introduction to Pattern Analysis
Ricardo Gutierrez-Osuna
Texas A&M University

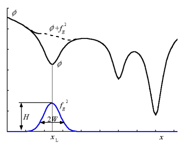
http://tr.im/yCaX

Problem Situations in Search: Review

- **Foothills *aka* Local Optima**

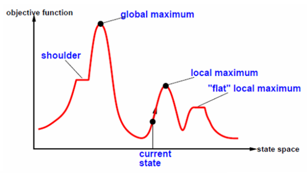


Wikipedia (2009).
http://tr.im/yDgf



Isshiki, M. (2000).
http://tr.im/yCt3

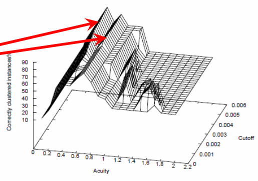
- **Plateaus (Plateaux) *aka* Lack of Gradient**



- **Single-Step Traps *aka* Ridges**

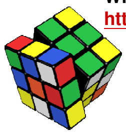
Ridges

Kansas State University
http://people.cis.ksu.edu/~welshamy/



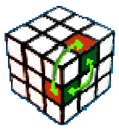
Solution 1: Macros Review

Wikipedia (2009). *Rubik's Cube.* http://en.wikipedia.org/wiki/Rubik%27s_Cube

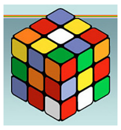


© 2007 Harris, D., Lee, J., & Mao, T.
http://tr.im/yCX7

Swapping edge cubelets:

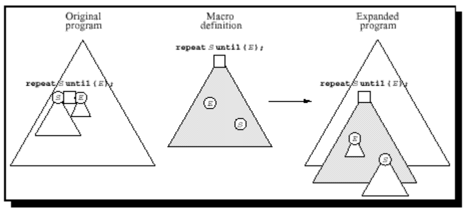


Rubik's Cube
© 1980 Rubik, E.
® Seven Towns, Ltd.



"How to solve the Rubik's Cube."
© 2009 Daum, N.
http://tr.im/yCSA
(Solution uses 7 macro steps such as the one shown at right.)

Ri U Fi Ui
BEFORE
AFTER
STEP 1: Solve the Upper Green Cross



© 2001 Schwartzbach, M., *et al.*
Basic Research in Computer Science (BRICS)
Aarhus University, Denmark
http://www.brics.dk/bigwig/refman/macro/

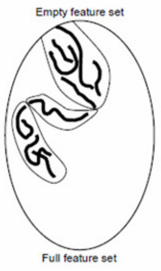
Solution 2A: Simulated Annealing (SA) Review

- **Simulated Annealing is a stochastic optimization method that derives its name from the annealing process used to re-crystallize metals**
  - During the annealing process in metals, the alloy is cooled down slowly to allow its atoms to reach a configuration of minimum energy (a perfectly regular crystal)
    - If the alloy is annealed too fast, such an organization cannot propagate throughout the material. The result will be a material with regions of regular structure separated by boundaries. These boundaries are potential fault-lines where fractures are most likely to occur when the material is stressed
  - The laws of thermodynamics state that, at temperature T, the probability of an increase in energy ΔE in the system is given by the expression

$$P(\Delta E) = e^{-\frac{\Delta E}{kT}}$$

    - where k is known as Boltzmann's constant
- **The SA algorithm is a straightforward implementation of these ideas**

```
1. Determine an annealing schedule T(i)
2. Create an initial solution Y(0)
3. While T(i)>T_MIN
   3a. Generate a new solution Y(i+1) which is a neighbor of Y(i)
   3b. Compute ΔE= [ J(Y(i)) - J(Y(i+1)) ]
   3b. If ΔE<0
          then
             always accept the move from Y(i) to Y(i+1)
          else
             accept the move with probability P=exp(-ΔE/T(i))
```



Introduction to Pattern Analysis
Ricardo Gutierrez-Osuna
Texas A&M University

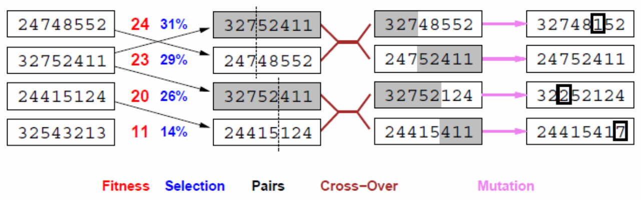
© 2005 R. Gutierrez-Osuna, Texas A&M University http://tr.im/yCaX

Solution 2B: Genetic Algorithm (GA) Review

Algorithm

1. Create an initial random population
2. Evaluate initial population
2. Repeat until convergence (or a number of generations)
   - 2a. Select the fittest individuals in the population
   - 2b. Perform crossover on the selected individuals to create offspring
   - 2c. Perform mutation on the selected individuals
   - 2d. Create the new population from the old population and the offspring
   - 2e. Evaluate the new population

= stochastic local beam search + generate successors from pairs of states



Constraint Satisfaction Problems (CSPs)

Standard search problem:
state is a "black box"—any old data structure
that supports goal test, eval, successor

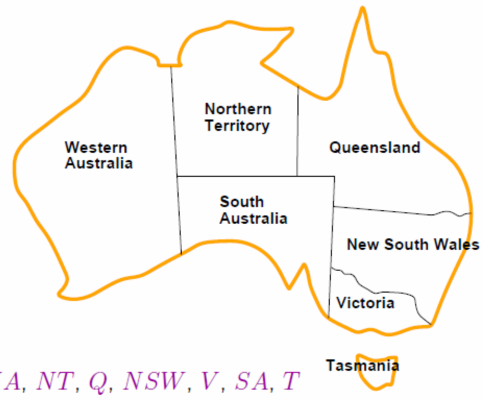
CSP:
state is defined by variables X_i with values from domain D_i

goal test is a set of constraints specifying
allowable combinations of values for subsets of variables

Simple example of a formal representation language

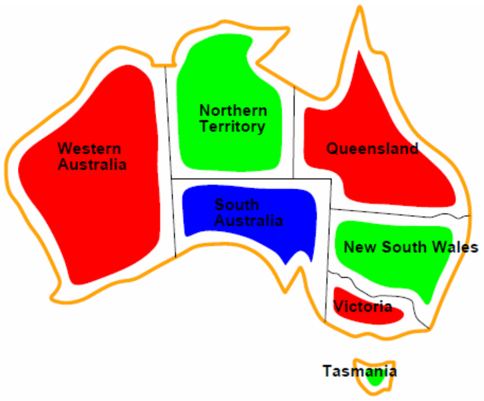
Allows useful general-purpose algorithms with more power than standard search algorithms

Example: Map Coloring [1]



Variables WA, NT, Q, NSW, V, SA, T
Domains $D_i = \{red, green, blue\}$
Constraints: adjacent regions must have different colors
e.g., $WA \neq NT$ (if the language allows this), or
$(WA, NT) \in \{(red, green), (red, blue), (green, red), (green, blue), \ldots\}$

Example: Map Coloring [2]

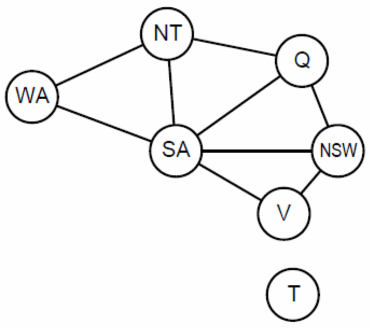


Solutions are assignments satisfying all constraints, e.g.,
$\{WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green\}$

Constraint Graph

Binary CSP: each constraint relates at most two variables

Constraint graph: nodes are variables, arcs show constraints



General-purpose CSP algorithms use the graph structure
to speed up search. E.g., Tasmania is an independent subproblem!

Varieties of CSPs

Discrete variables

finite domains; size d $\Rightarrow$ $O(d^n)$ complete assignments

- ◇ e.g., Boolean CSPs, incl. Boolean satisfiability (NP-complete)

infinite domains (integers, strings, etc.)

- ◇ e.g., job scheduling, variables are start/end days for each job
- ◇ need a constraint language, e.g., $StartJob_1 + 5 \leq StartJob_3$
- ◇ linear constraints solvable, nonlinear undecidable

Continuous variables

- ◇ e.g., start/end times for Hubble Telescope observations
- ◇ linear constraints solvable in poly time by LP methods

Varieties of Constraints

Unary constraints involve a single variable,
e.g., $SA \neq green$

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 (soft constraints), e.g., red is better than $green$
often representable by a cost for each variable assignment
$\rightarrow$ constrained optimization problems

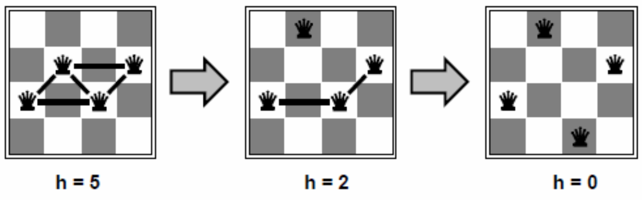
Example: N-Queens

States: 4 queens in 4 columns ($4^4 = 256$ states)

Operators: move queen in column

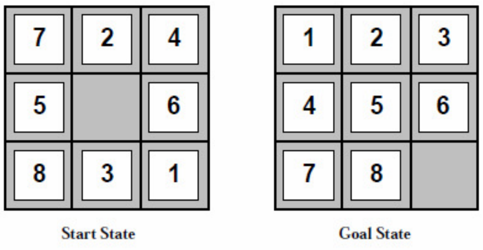
Goal test: no attacks

Evaluation: $h(n)$ = number of attacks



Example: 8-Puzzle

Last seen in Chapters 3 – 4 (Uninformed & Informed Search)
How can we make this a CSP?

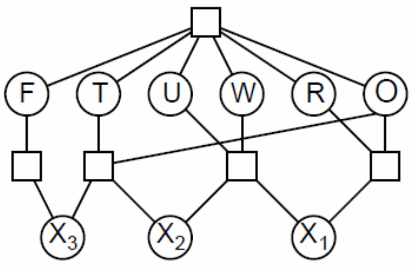


states??: integer locations of tiles (ignore intermediate positions)
actions??: move blank left, right, up, down (ignore unjamming etc.)
goal test??: = goal state (given)
path cost??: 1 per move

[Note: optimal solution of n-Puzzle family is NP-hard]

Example: Cryptarithmetic

```
    T W O
+   T W O
 F  O U R
```



Variables: $F\ T\ U\ W\ R\ O\ X_1\ X_2\ X_3$
Domains: $\{0,1,2,3,4,5,6,7,8,9\}$
Constraints
 alldiff(F,T,U,W,R,O)
 $O+O=R+10\cdot X_1$, etc.

Example: Missionaries and Cannibals

- 3 Missionaries (A, B, C)
- 3 Cannibals (a, b, c)
- Start State: {ABC abc, ∅}
- Goal State: {∅, ABC abc}
- Constraints
  - Boat: can have 1 or 2 occupants
  - Bank: must have more Ms than Cs (otherwise Cs eat Ms)
- State Space Search
  - 13 states shown
  - 12 transitions
- CSP Representation: Exercise

Shortest Solution (Non-Unique)

Trip number	Starting bank	Travel	Ending bank
(start)	Aa Bb Cc		
1	Bb Cc	Aa →	
2	Bb Cc	← A	a
3	A B C	bc →	a
4	A B C	← a	b c
5	Aa	BC →	b c
6	Aa	← Bb	Cc
7	a b	AB →	Cc
8	a b	← c	A B C
9	b	a c →	A B C
10	b	← B	Aa Cc
11		Bb →	Aa Cc
(finish)			Aa Bb Cc

Wikipedia (2009).
Missionaries and Cannibals problem.
http://tr.im/yDoJ

Example: Farmer, Fox, Goose, and Grain

- Agent: Farmer F
- Objects
  - A: Predator (Fox, Wolf, Panther, *etc.*)
  - B: Meat animal (Goose, Sheep, Pig, *etc.*)
  - C: Crop (Grain, Beans, Cabbage, Corn, etc.)
- Initial State: {FABC, ∅}
- Goal State: {∅, FABC}
- CSP Representation Sketch
  - Boat: can have 1 objects
  - Bank: must not contain AB or BC
- State Space Representation: Exercise

Shortest Solution (Non-Unique)

1. Bring goose over
2. Return
3. Bring fox or beans over
4. Bring goose back
5. Bring beans or fox over
6. Return
7. Bring goose over

Wikipedia (2009).
Farmer, Goose, and Bag of Beans Puzzle.
http://tr.im/yDom

Real-World CSPs

Assignment problems
    e.g., who teaches what class

Timetabling problems
    e.g., which class is offered when and where?

Hardware configuration

Spreadsheets

Transportation scheduling

Factory scheduling

Floorplanning

Notice that many real-world problems involve real-valued variables

Standard Search Formulation: Incremental

Let's start with the straightforward, dumb approach, then fix 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.
    $\Rightarrow$ fail if no legal assignments (not fixable!)

◇ Goal test: the current assignment is complete

1) This is the same for all CSPs!
2) Every solution appears at depth n with n variables
    $\Rightarrow$ use depth-first search
3) Path is irrelevant, so can also use complete-state formulation
4) $b = (n - \ell)d$ at depth ℓ, hence $n!d^n$ leaves!!!!

Backtracking Search [1]: Description

Variable assignments are commutative, i.e.,
$[WA = red$ then $NT = green]$ same as $[NT = green$ then $WA = red]$

Only need to consider assignments to a single variable at each node
$\Rightarrow$ $b = d$ 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

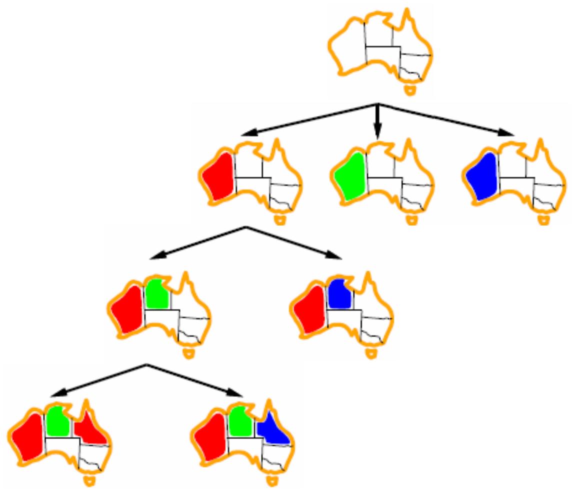
Can solve n-queens for $n \approx 25$

Backtracking Search [2]: Algorithm

```
function BACKTRACKING-SEARCH(csp) returns solution/failure
   return RECURSIVE-BACKTRACKING({ }, csp)

function RECURSIVE-BACKTRACKING(assignment, csp) returns soln/failure
   if assignment is complete then return assignment
   var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp], assignment, csp)
   for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
      if value is consistent with assignment given CONSTRAINTS[csp] then
         add {var = value} to assignment
         result ← RECURSIVE-BACKTRACKING(assignment, csp)
         if result ≠ failure then return result
         remove {var = value} from assignment
   return failure
```

Backtracking Example: Graph 3-Coloring



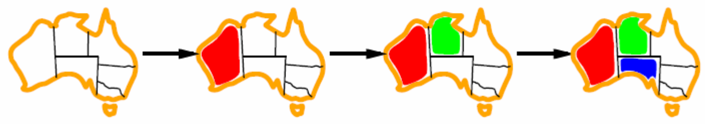
Improving Backtracking Efficiency

General-purpose methods can give huge gains in speed:

1. Which variable should be assigned next?
2. In what order should its values be tried?
3. Can we detect inevitable failure early?
4. Can we take advantage of problem structure?

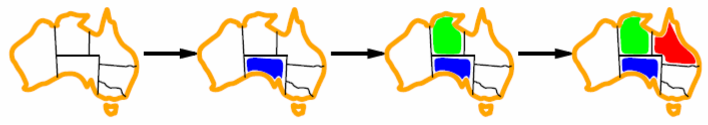
Variable Selection: Minimum Remaining Values

Minimum remaining values (MRV):
choose the variable with the fewest legal values



Tie-breaker among MRV variables

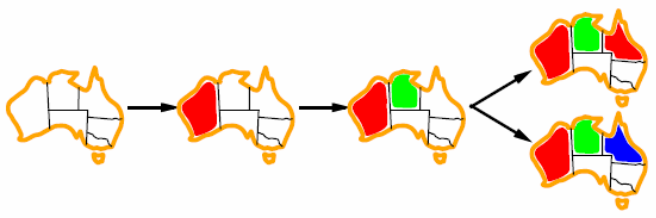
Degree heuristic:
choose the variable with the most constraints on remaining variables



Based on slides © 2004 S. Russell & P. Norvig. Reused with permission.

Value Selection (Given Variable): Least Constraining Variable

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



Allows 1 value for SA

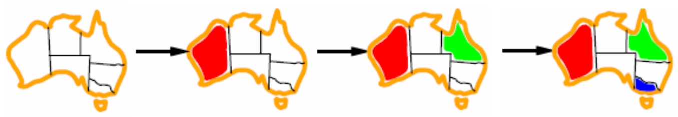
Allows 0 values for SA

Combining these heuristics makes 1000 queens feasible

Based on slide © 2004 S. Russell & P. Norvig. Reused with permission.

Forward Checking

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



WA	NT	Q	NSW	V	SA	T
🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦
🟥	🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟩🟦	🟥🟩🟦
🟥	🟦	🟩	🟥🟦	🟥🟩🟦	🟦	🟥🟩🟦
🟥	🟦	🟩	🟥	🟦		🟥🟩🟦

Constraint Propagation

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

WA	NT	Q	NSW	V	SA	T
🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦
🟥	🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟩🟦	🟥🟩🟦
🟥	🟦	🟩	🟥🟦	🟥🟩🟦	🟦	🟥🟩🟦

NT and SA cannot both be blue!

Constraint propagation repeatedly enforces constraints locally

Terminology

- Problem Situations in Heuristic Search: Foothills, Plateaus, Ridges
- Solutions: Macro Operators (Macrops), Global Optimization (GA/SA)
- Constraint Satisfaction Problems (CSPs)
  - Variables: values range over domain
  - Constraints: relations that must hold
    - ⇨ *n*-ary ($n = 1$: unary; $n = 2$: binary; $n = 3$: ternary; $n = 4$: quaternary; *etc.*)
    - ⇨ Denoted (represented) by predicates over *n* variables
  - CSP: collection of variables, constraints (compare: node in graph search)
  - Partial instantiation: set of variable-value bindings
- CSPs: *n*-Queens, 8-Puzzle, Cryptarithmetic, Missionaries, Farmer
- CSP Techniques
  - Variable selection heuristic: Minimum Remaining Values (MRV)
  - Value selection heuristic: Least Constraining Value (LCV)
  - Forward checking algorithm: eliminates values made illegal by commitments
- Detailed CSP Example: Graph Coloring
  - Graph coloring: 2, 3, or 4-color specified graph G = (V, E)
  - Types of graphs: bipartite, complete, complete bipartite, planar

Summary Points

- Problems in Heuristic Search: Foothills, Plateaus, Ridges
- Solution: Macros and GO
  - Macros: encode reusable sequences of steps
  - Global optimization (GO): SA, GA
- CSP Examples
  - *n*-queens: standard form (n = 8), larger instances (100, 1000, etc.)
  - 8-Puzzle, Cryptarithmetic
  - River crossing puzzles
    - ⇨ Missionaries and cannibals (*aka* "jealous husbands")
    - ⇨ Farmer, fox, goose, and grain (or beans)
- First Algorithm: Backtracking Search with Heuristics
  - Minimum Remaining Values (for choosing variable)
  - Least Constraining Value (for choosing value, given variable)
- Second Algorithm: Forward Checking with Constraint Propagation
- Detailed CSP Example: 3-Coloring Australian Map