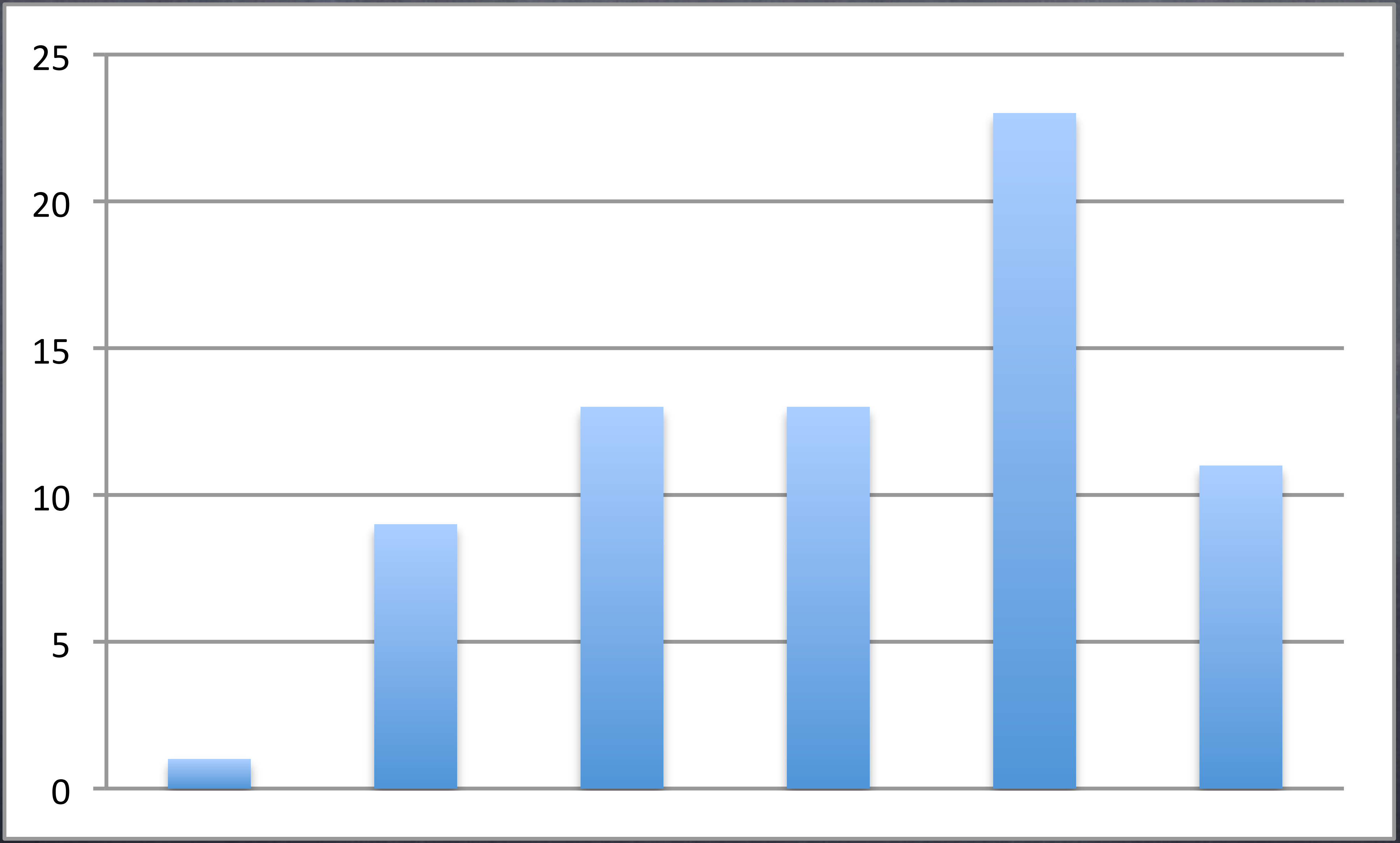
CSC242: Intro to AI

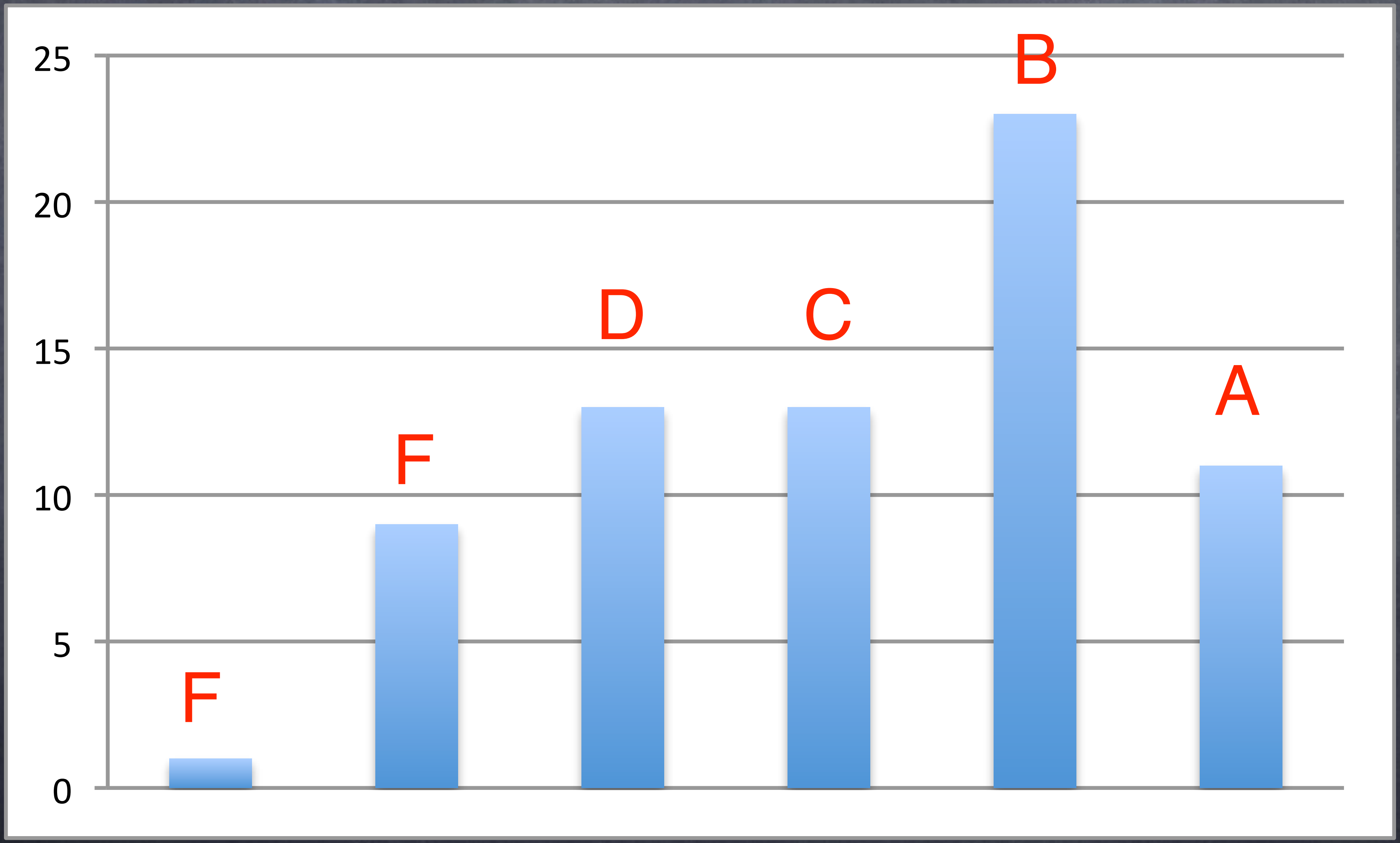
Lecture 7

Games of Imperfect Knowledge & Constraint Satisfaction

What is This?



Quiz 1



Moral

- Many people cannot learn from lectures
- Do the homework!
  - If you do, exams will be easy
  - If you don't, exams will be impossible
- First exam: February 20

The Road Ahead

- Complete calendar for the semester now online
  - Topics
  - Exam dates
  - Project dates
- 3 programming projects
- 3 in-class exams + final exam

Othello

- Phase I due Feb 25
- Project page updated, (re)check details!
- Generating legal moves is not trivial!
    - A legal move must capture some pieces!
- My own solution: 125 lines of Python
    - == 125 lines of C == 250 lines of Java

Stochastic Games of Perfect Information

Stochastic Games of Perfect Information

- Examples:
  - Backgammon
  - Roulette
  - Candyland
  - Parcheesi
- Why stochastic?
- Why perfect information?

Stochastic Games of Perfect Information

- Examples:
  - Backgammon
  - Roulette
  - Candyland
  - Parcheesi

- Why stochastic? Contains a random element
- Why perfect information?

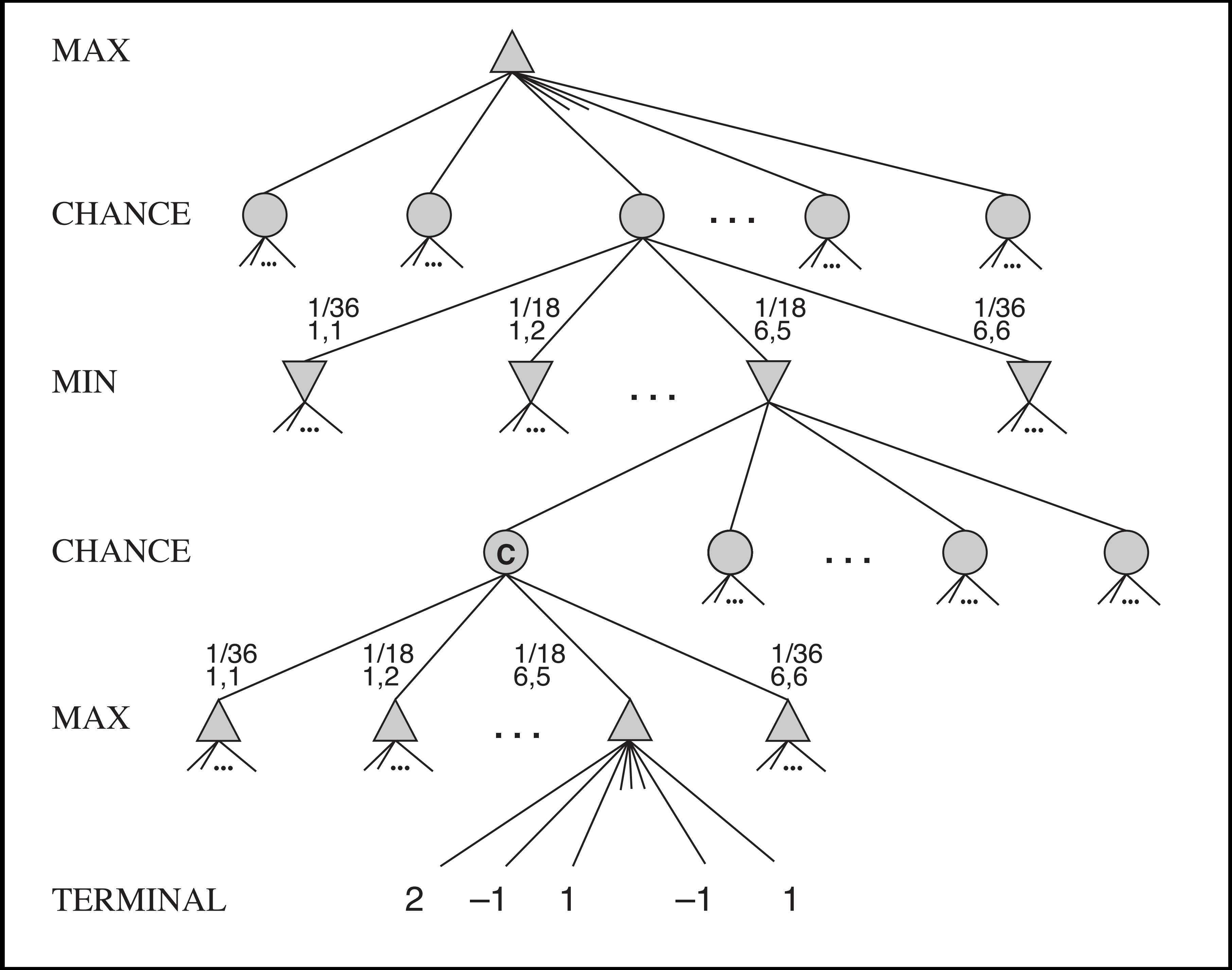
Stochastic Games of Perfect Information

- Examples:
  - Backgammon
  - Roulette
  - Candyland
  - Parcheesi

- Why stochastic? Contains a random element
- Why perfect information? No hidden state!

Expecti-Minimax

- Same as MINIMAX for MIN and MAX nodes
- Same backing up utilities from terminal nodes
- Take expectation over chance nodes
  - Weighted average of possible outcomes


MAX
CHANCE
MIN
CHANCE
MAX
TERMINAL
1/36
1,1
1/18
1,2
1/18
6,5
1/36
6,6
C
1/36
1,1
1/18
1,2
1/18
6,5
1/36
6,6
2
–1
1
–1
1

Expecti-Minimax

$$\textsc{EMinimax}(s) = \begin{cases} \textsc{Utility}(s) & \text{if } \textsc{Terminal-Test}(s) \\ \max_a \textsc{EMinimax}(\textsc{Result}(S,a)) & \text{if } \textsc{Player}(s) = \textsc{max} \\ \min_a \textsc{EMinimax}(\textsc{Result}(S,a)) & \text{if } \textsc{Player}(s) = \textsc{min} \\ \sum_r P(r)\textsc{EMinimax}(\textsc{Result}(S,r)) & \text{if } \textsc{Player}(s) = \textsc{chance} \end{cases}$$

Partial Observability

- Some of the state of the world is hidden (unobservable)

Partially-Observable Games

- Some of the state of the game is hidden from the player(s)
- Interesting because:
  - Valuable real-world games like poker
  - Partial observability arises all the time in real-world problems

Partially-Observable Games

- Deterministic partial observability
  - Opponent has hidden state
  - No element of randomness
  - Examples?

Partially-Observable Games

- Deterministic partial observability
  - Opponent has hidden state
  - Battleship, Stratego

Partially-Observable Games

- Deterministic partial observability
  - Opponent has hidden state
  - Battleship, Stratego
- Stochastic partial observability
  - Hidden information is random
  - Examples?

Stochastic Partially Observable Games

Hand	Frequency	Approx. Probability	Approx. Cumulative	Approx. Odds	Mathematical expression of absolute frequency
Royal flush 	4	0.000154%	0.000154%	649,739 : 1	$\binom{4}{1}$
Straight flush (excluding royal flush) 	36	0.00139%	0.00154%	72,192.33 : 1	$\binom{10}{1}\binom{4}{1}-\binom{4}{1}$
Four of a kind 	624	0.0240%	0.0256%	4,164 : 1	$\binom{13}{1}\binom{12}{1}\binom{4}{1}$
Full house 	3,744	0.144%	0.170%	693.2 : 1	$\binom{13}{1}\binom{4}{3}\binom{12}{1}\binom{4}{2}$
Flush (excluding royal flush and straight flush) 	5,108	0.197%	0.367%	507.8 : 1	$\binom{13}{5}\binom{4}{1}-\binom{10}{1}\binom{4}{1}$
Straight (excluding royal flush and straight flush) 	10,200	0.392%	0.76%	253.8 : 1	$\binom{10}{1}\binom{4}{1}^5-\binom{10}{1}\binom{4}{1}$
Three of a kind 	54,912	2.11%	2.87%	46.3 : 1	$\binom{13}{1}\binom{4}{3}\binom{12}{2}\binom{4}{1}^2$
Two pair 	123,552	4.75%	7.62%	20.03 : 1	$\binom{13}{2}\binom{4}{2}^2\binom{11}{1}\binom{4}{1}$
One pair 	1,098,240	42.3%	49.9%	1.36 : 1	$\binom{13}{1}\binom{4}{2}\binom{12}{3}\binom{4}{1}^3$
No pair / High card 	1,302,540	50.1%	100%	.995 : 1	$\left[\binom{13}{5}-10\right]\left[\binom{4}{1}^5-4\right]$
Total	**2,598,960**	**100%**	**100%**	**1 : 1**	$\binom{52}{5}$

Weighted Minimax

- For each possible deal s:
  - Assume s is the actual situation
  - Compute Minimax or H-Minimax value of s
  - Weight value by probability of s
- Take move that yields highest expected value over all the possible deals

Weighted Minimax

$$\operatorname*{argmax}_a \sum_s P(s)\textsc{Minimax}(\textsc{Result}(s, a))$$

Weighted Minimax

$$\underset{a}{\operatorname{argmax}} \sum_{s} P(s)\,\textsc{Minimax}(\textsc{Result}(s,a))$$

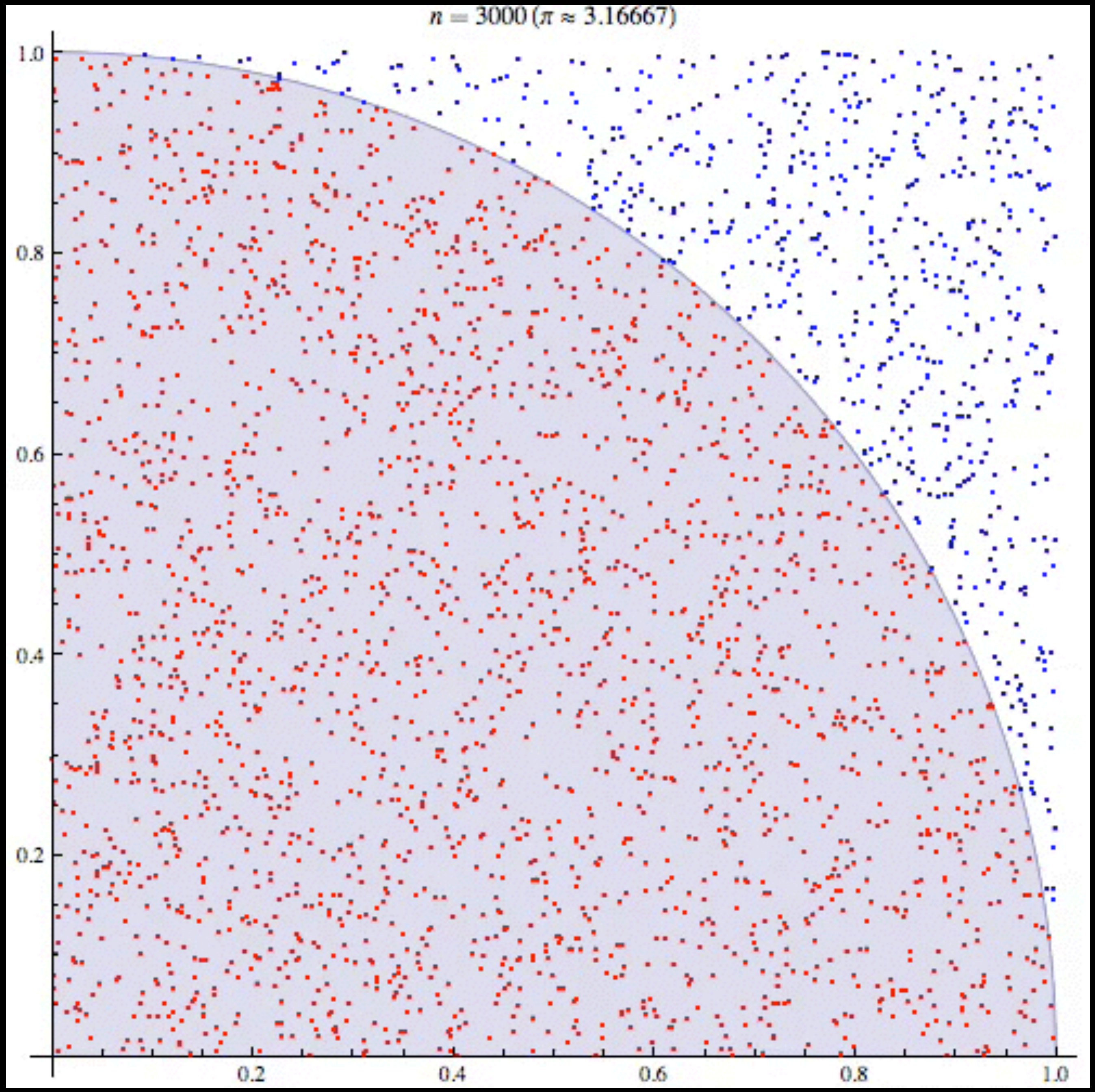
2-Player Hearts: $\binom{52-13}{13} = 8 \times 10^{9}$

4-Player Hearts: $\binom{39}{13}\binom{26}{13}\binom{13}{13} = 8 \times 10^{16}$

4-Player Poker: $\binom{47}{5}\binom{42}{5}\binom{37}{5} = 1 \times 10^{17}$

Monte Carlo Methods

- Use a "representative" sample to approximate a large, complex distribution



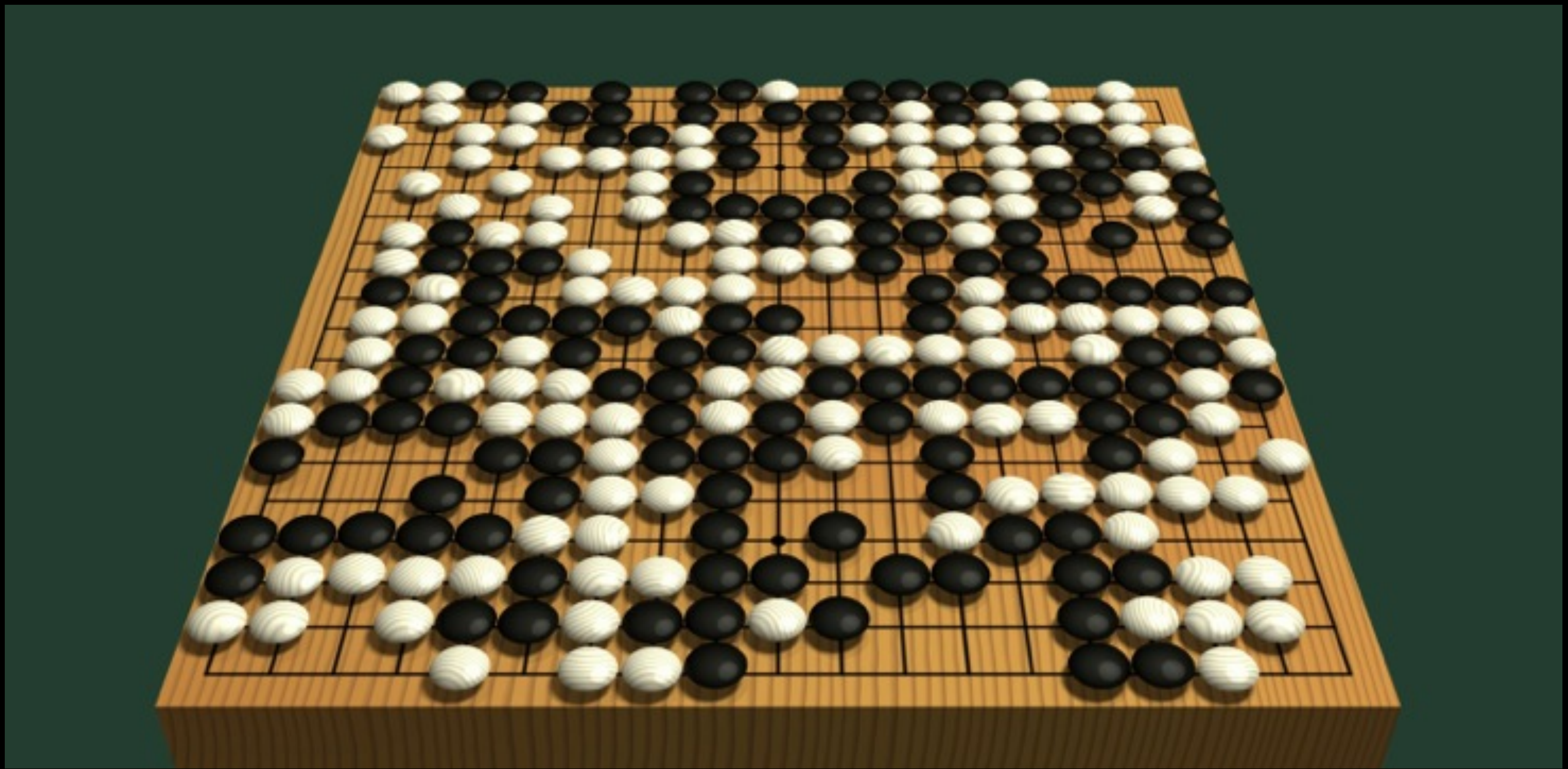
Monte-Carlo Minimax

$$\operatorname*{argmax}_a \frac{1}{N} \sum_{i=1}^{N} \textsc{Minimax}(\textsc{Result}(s_i, a))$$

- Can also sample during minimax search
  - Equivalently: expand a random sample of children at each level
- Used in champion card playing programs
  - Bridge, Poker

Monte Carlo MiniMax

- Useful even for deterministic games of perfect information that have very high branching factors!



Summary

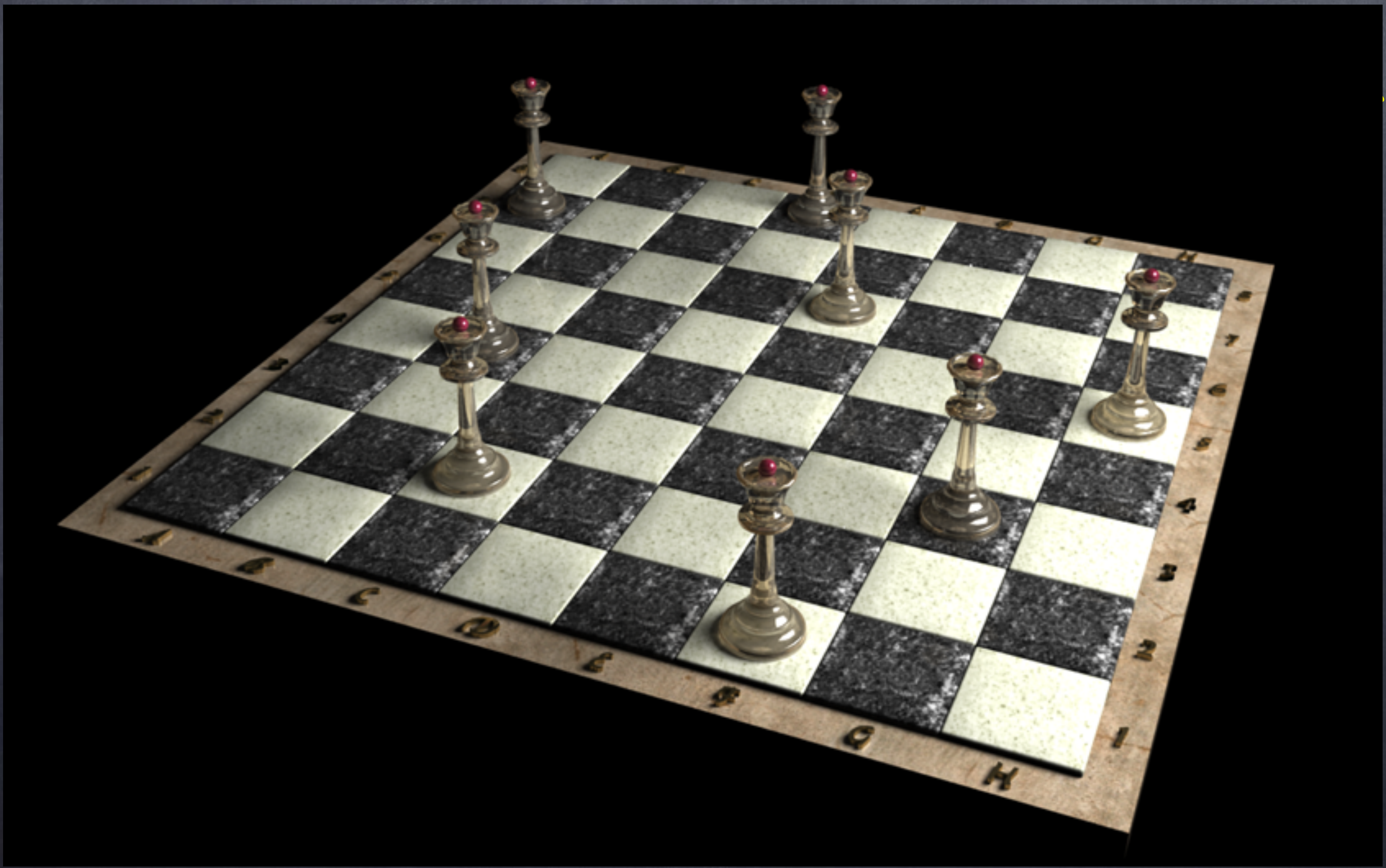
- Stochastic games
  - Expecti-MINIMAX: Compute expected MINIMAX value over chance nodes
- Partially observable games
  - Weighted MINIMAX: Compute expected value over possible hidden states
  - When tree becomes too large, sample branches rather than explore exhaustively

Constraint Satisfaction

What is Constraint Satisfaction?

- In most of the search problems we have discussed up to now, a solution corresponds to a path or the initial step in a path through a state space
  - Route-finding
  - Solving the 8 Puzzle
  - Game playing

What is Constraint Satisfaction?



The Problem With State-Space Search

- State representation is specific to a given problem (or domain of problems)
- Functions on states (successor generation, goal test) are specific to the state representation
- Heuristic functions are both problem-specific and dependent on the state representation
- Many design choices, many opportunities for coding errors

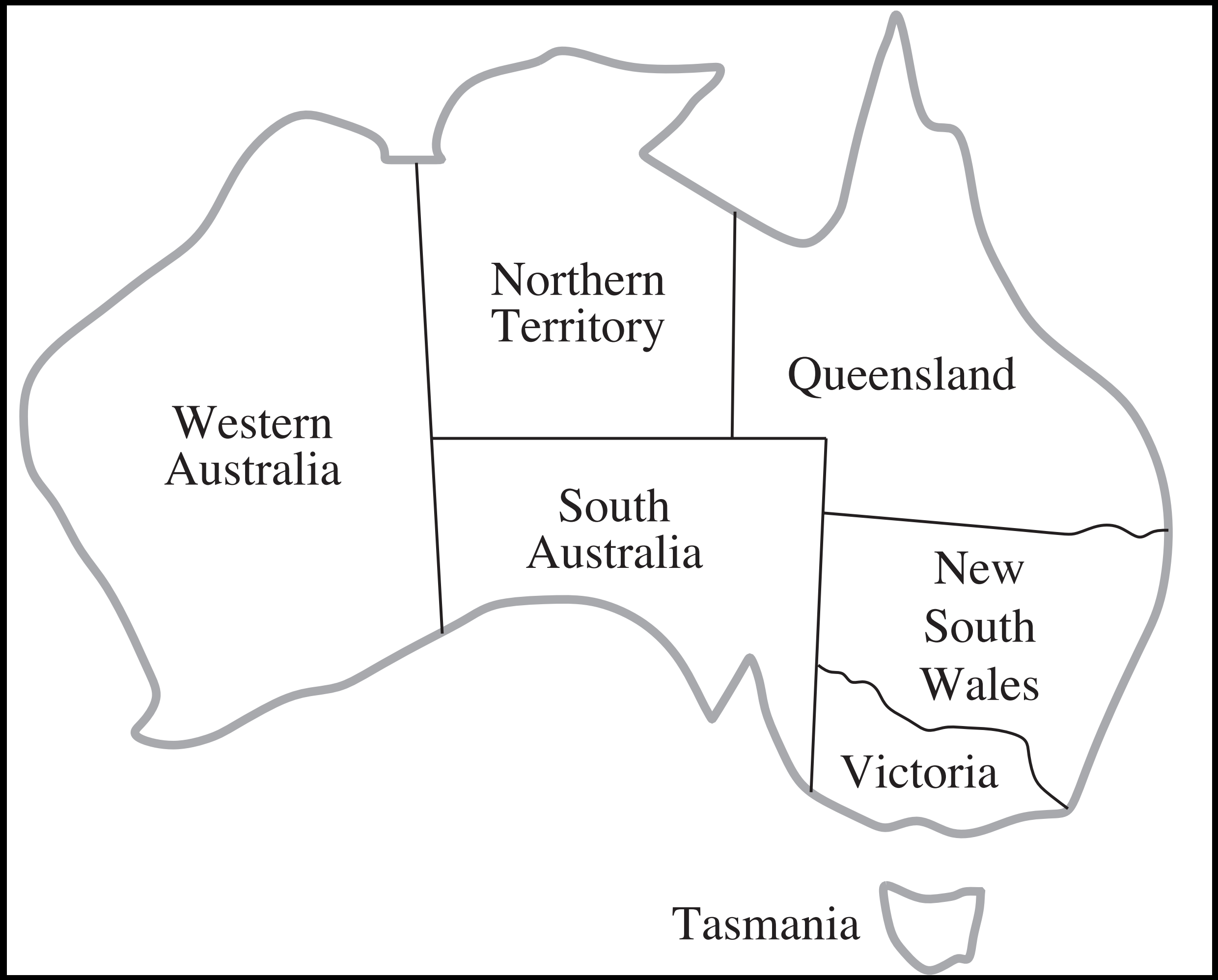
The CSP Approach

- Impose a structure on the representation of states
- Using that representation, successor generation and goal tests are problem- and domain-independent
- Can also develop effective problem- and domain-independent heuristics

Bottom Line

Represent State This Way → Write No Code! → No Bugs!

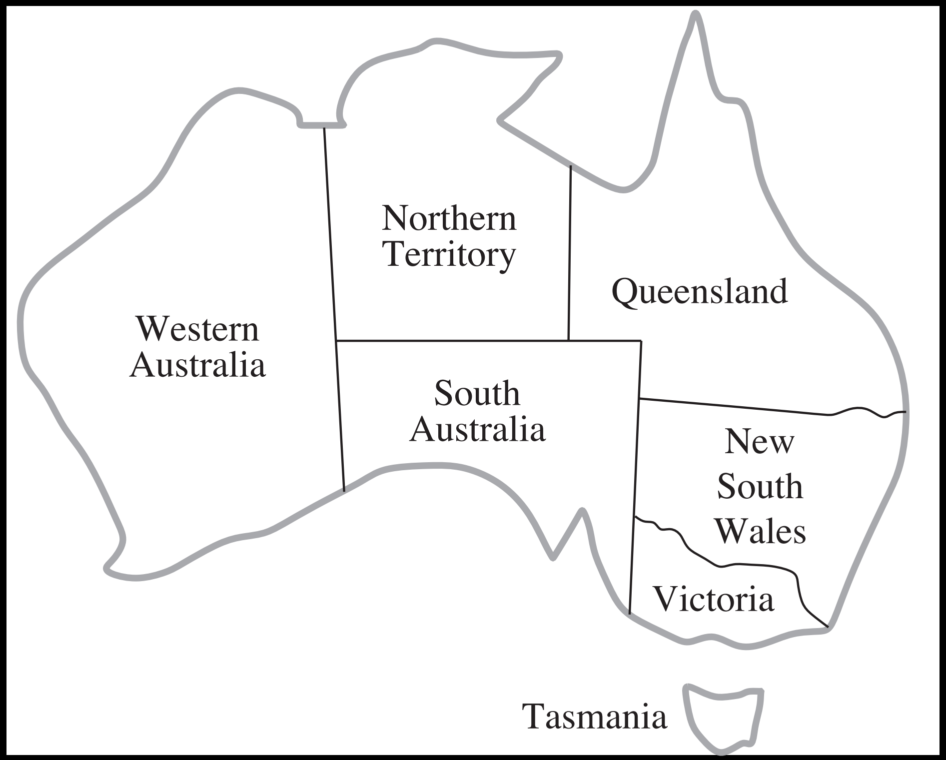
Example



Assign a color to each region such that no two neighboring regions have the same color

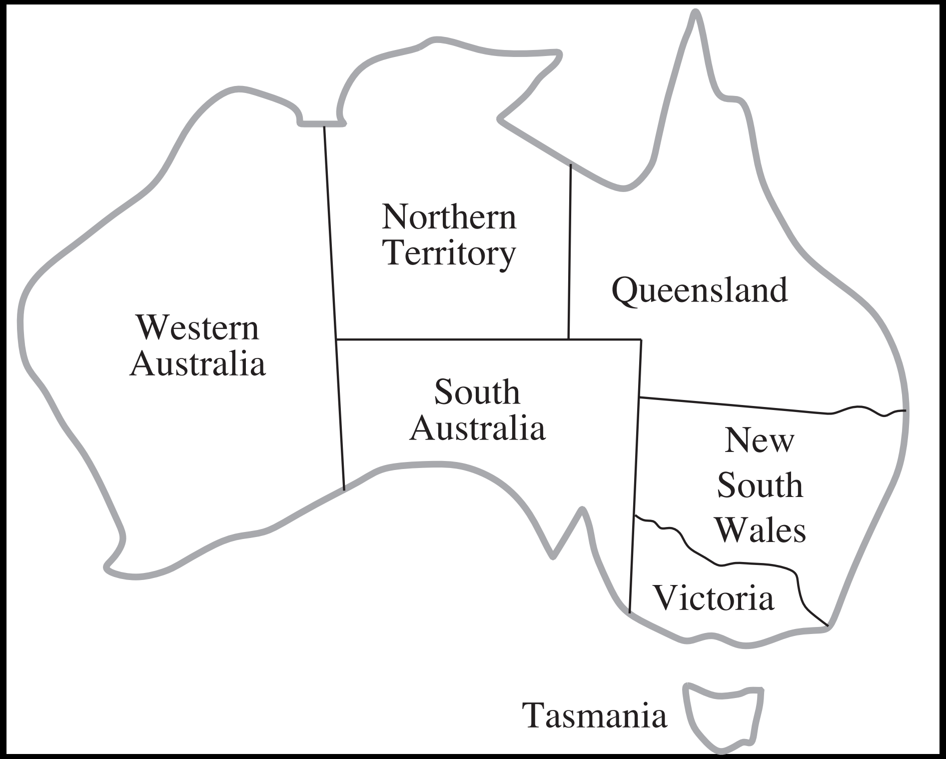
Color WA, NT, Q, NSW, V, SA, T

enum Color = red, green, blue



Color WA, NT, Q, NSW, V, SA, T

enum Color = red, green, blue

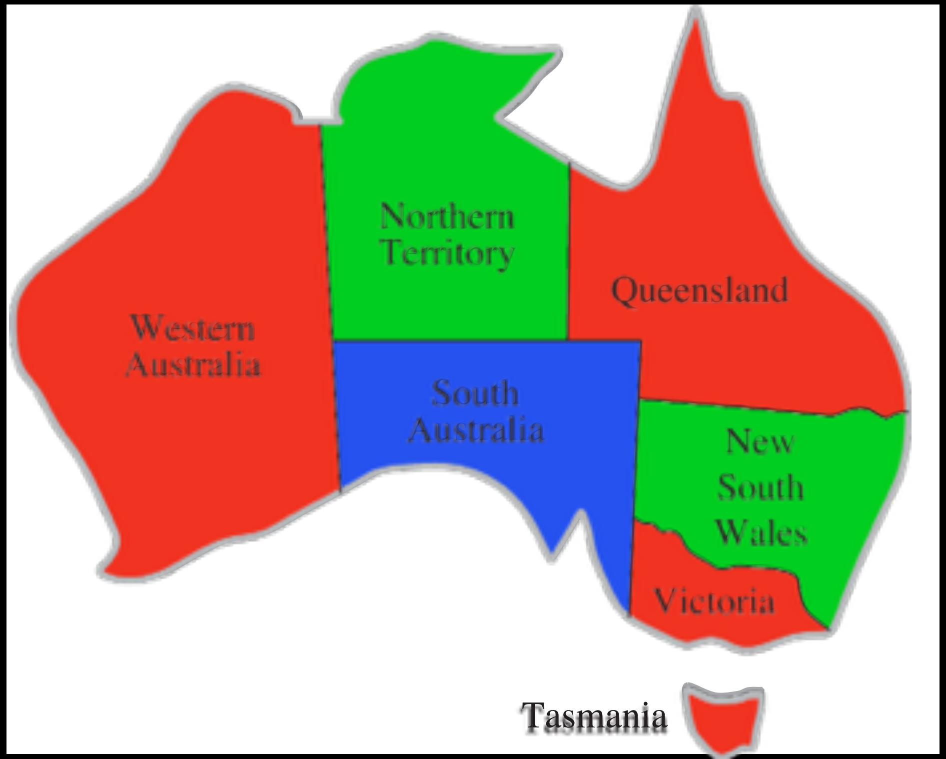


State: assignment of colors to regions

Successor function: pick an unassigned region and assign it a color

Goal test: All regions assigned and no adjacent regions have the same color

Color WA, NT, Q, NSW, V, SA, T



enum Color = red, green, blue

WA=red, NT=green, Q=red, NSW=green
V=red, SA=blue, T=red

Constraint Satisfaction Problem (CSP)

X: Set of variables $\{ X_1, ..., X_n \}$

D: Set of domains $\{ D_1, ..., D_n \}$
Each domain D_i = set of values $\{ v_1, ..., v_k \}$

C: Set of constraints $\{ C_1, ..., C_m \}$

Australia Map CSP

X: { X_i } = { WA, NT, Q, NSW, V, SA, T }

D: Each D_i = { red, green, blue }

C: { SA≠WA, SA≠NT, SA≠Q, SA≠NSW,
SA≠V, WA≠NT, NT≠Q, Q≠NSW,
VSW≠V }

More CSP Terminology

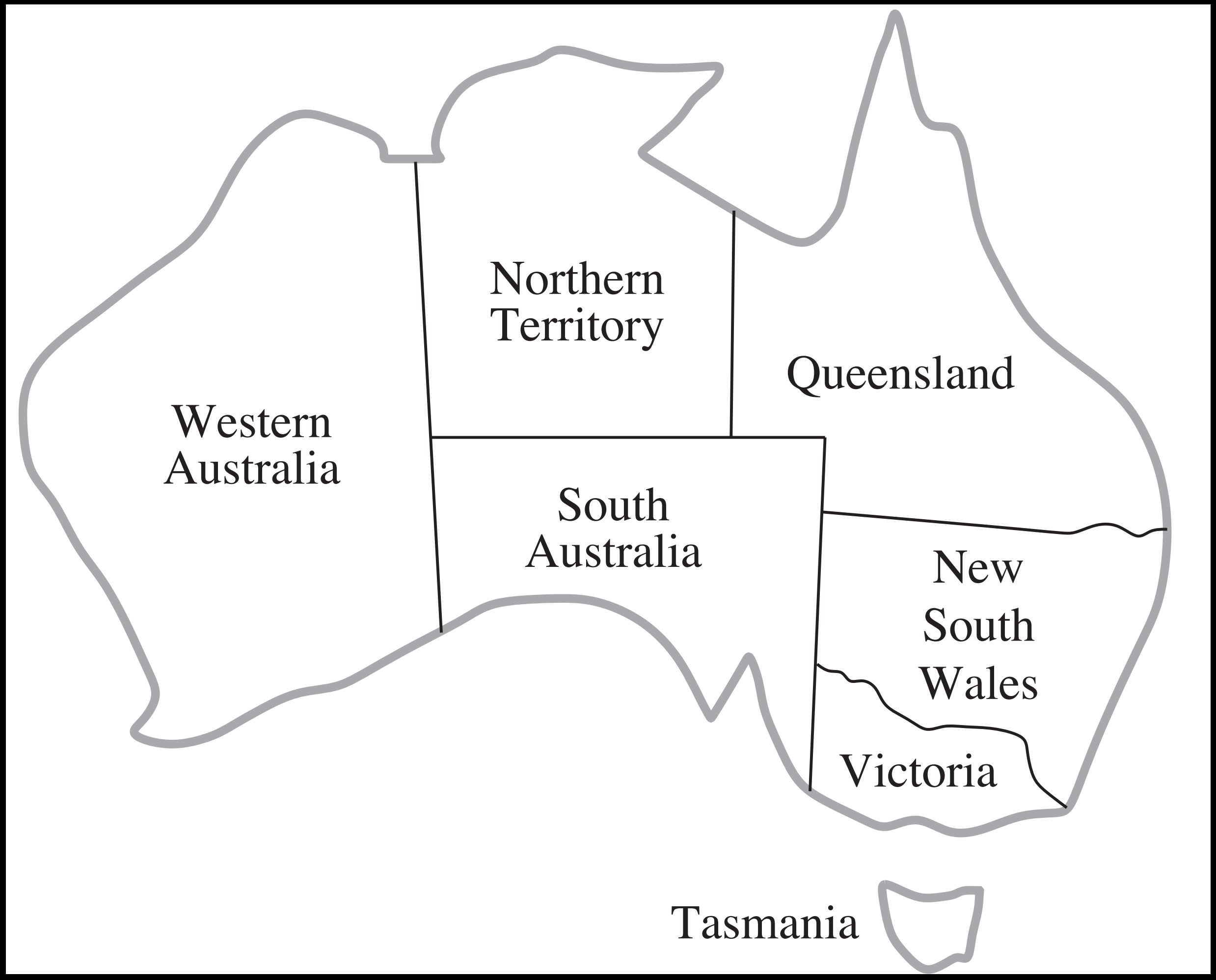
- Assignment: { $X_i = v_i$, $X_j = v_j$, ... }
- Consistent: does not violate any constraints
- Partial: some variables are unassigned
- Complete: every variable is assigned
- Solution: consistent, complete assignment

Constraints

- Unary constraint: one variable
  - e.g., NSW $\neq$ red, X_i is even, $X_i = 2$
- Binary constraint: two variables
  - e.g., NSW $\neq$ WA, $X_i > X_j$, $X_i+X_j = 2$
- "Global" constraint: more than two vars
  - e.g., X_i is between X_j and X_k, AllDiff(X_i,X_j,X_k)
  - Can be reduced to set of binary constraints (possibly inefficiently)

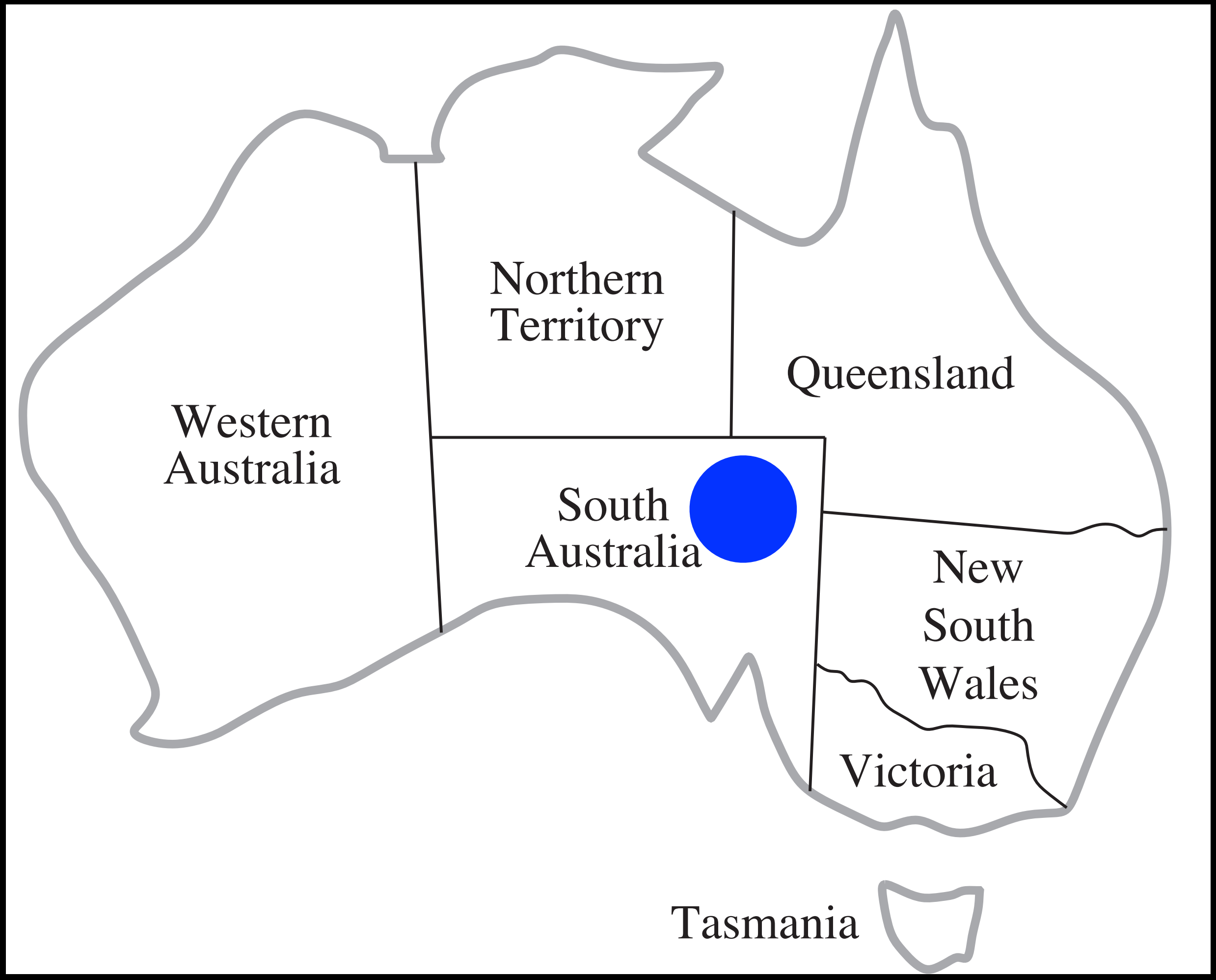
- Faster to search (solve)
- Problem-independent (no code!)
- Constraint propagation

WA	R, G, B
NT	R, G, B
SA	R, G, B
Q	R, G, B
NSW	R, G, B
V	R, G, B
T	R, G, B



Possibilities:
$3^7 = 2,187$

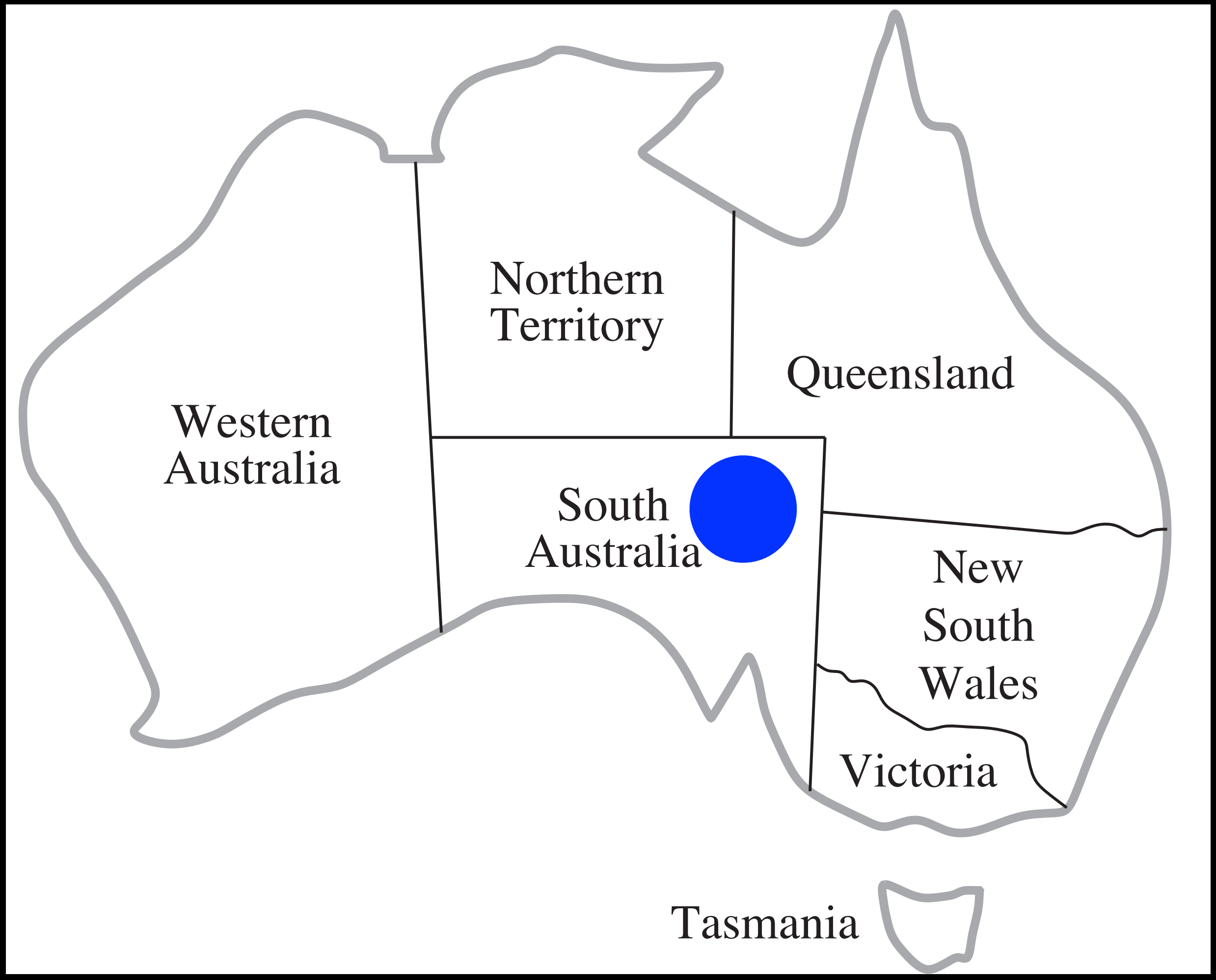
WA	R, G, B
NT	R, G, B
SA	B
Q	R, G, B
NSW	R, G, B
V	R, G, B
T	R, G, B



Make choice: color SA blue

Remaining possibilities:
$3^6 = 729$

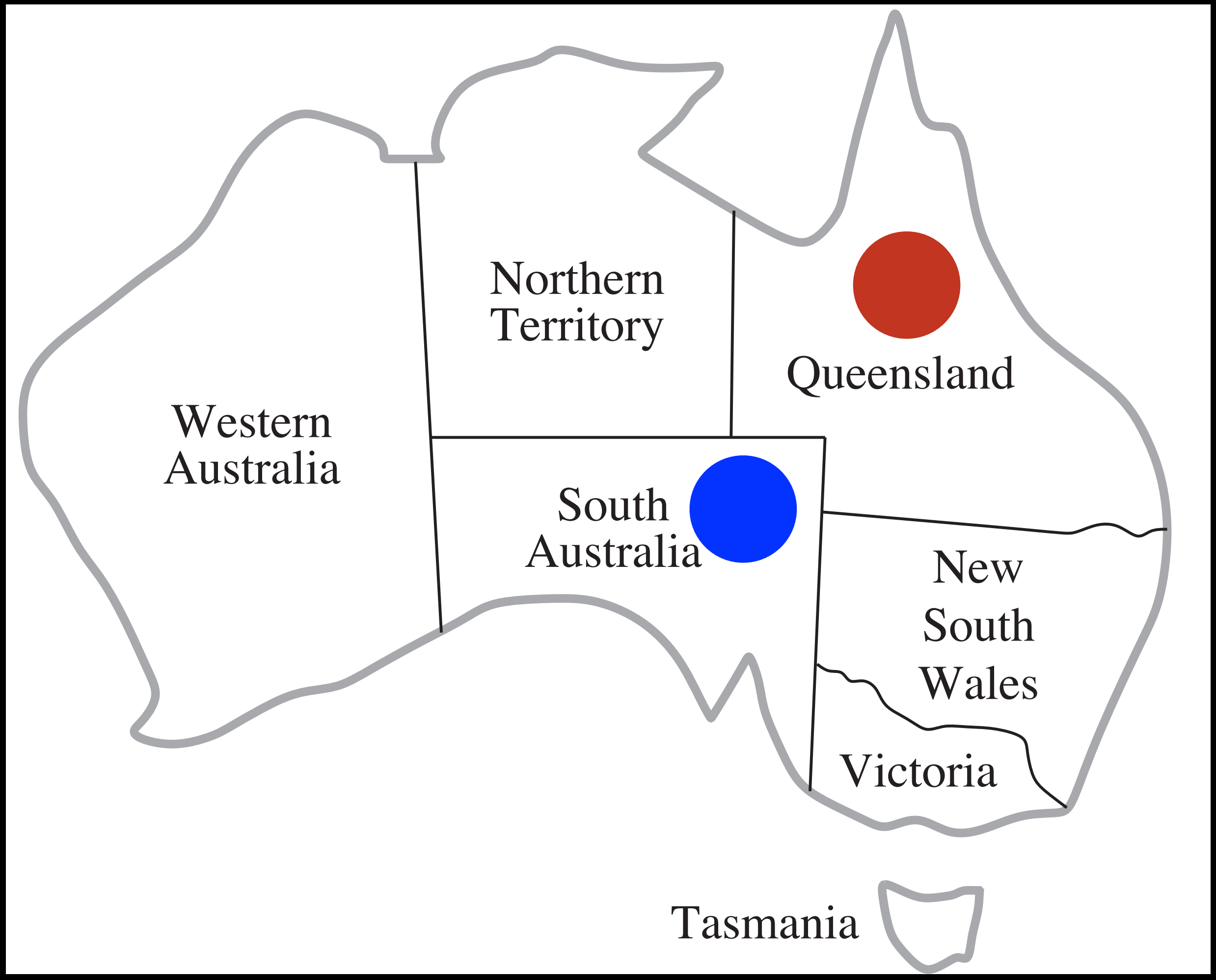
WA	R, G
NT	R, G
SA	B
Q	R, G
NSW	R, G
V	R, G
T	R, G, B



Simplify: remove B from adjacent regions

Remaining possibilities:
$2^5 \times 3 = 96$

WA	R, G
NT	R, G
SA	B
Q	R
NSW	R, G
V	R, G
T	R, G, B

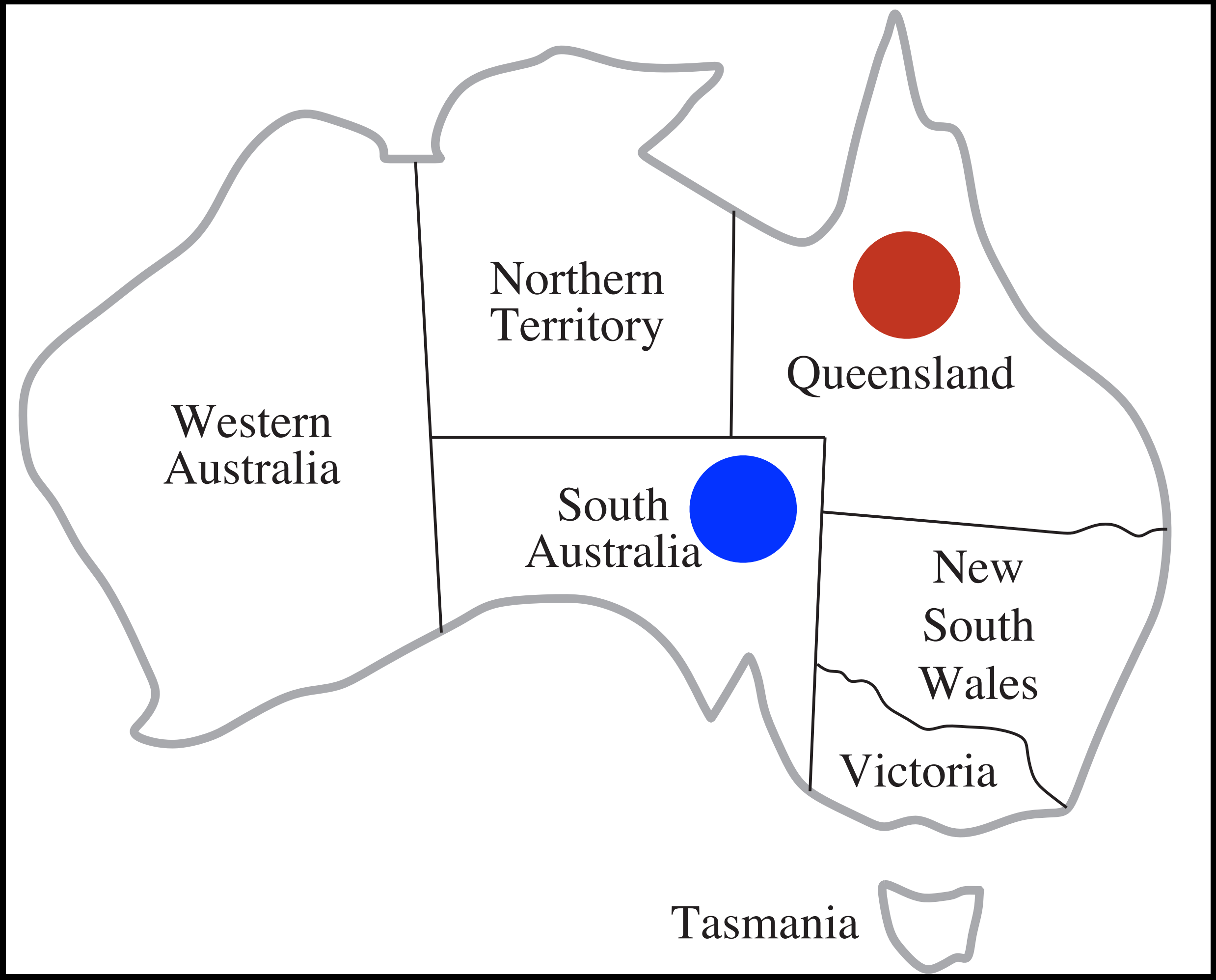


Make choice: color Q red

Remaining possibilities:

$2^4 \times 3 = 48$

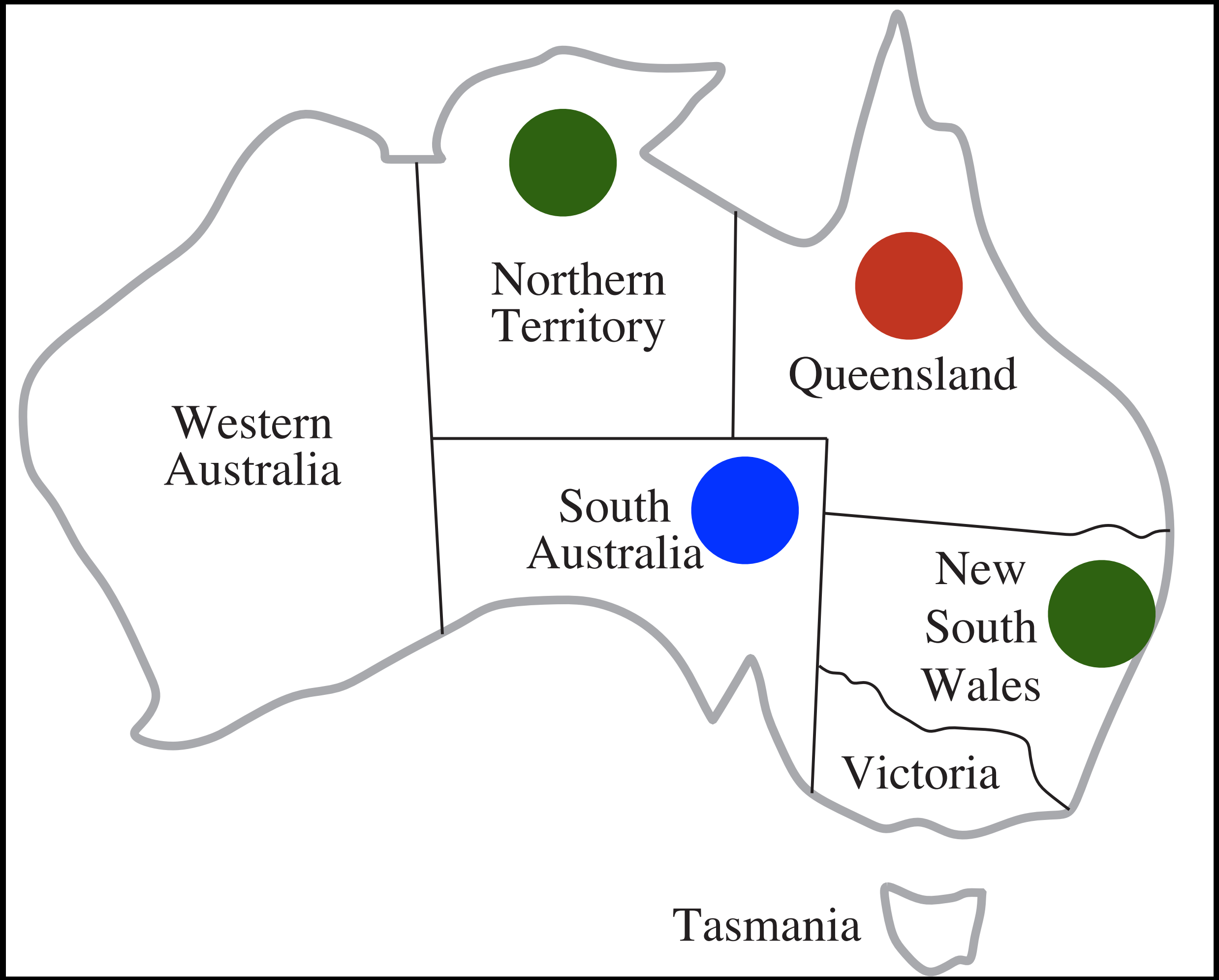
WA	R, G
NT	G
SA	B
Q	R
NSW	G
V	R, G
T	R, G, B



Simplify: remove R from adjacent regions

Remaining possibilities:
$2^2 \times 3 = 12$

WA	R, G
NT	G
SA	B
Q	R
NSW	G
V	R, G
T	R, G, B

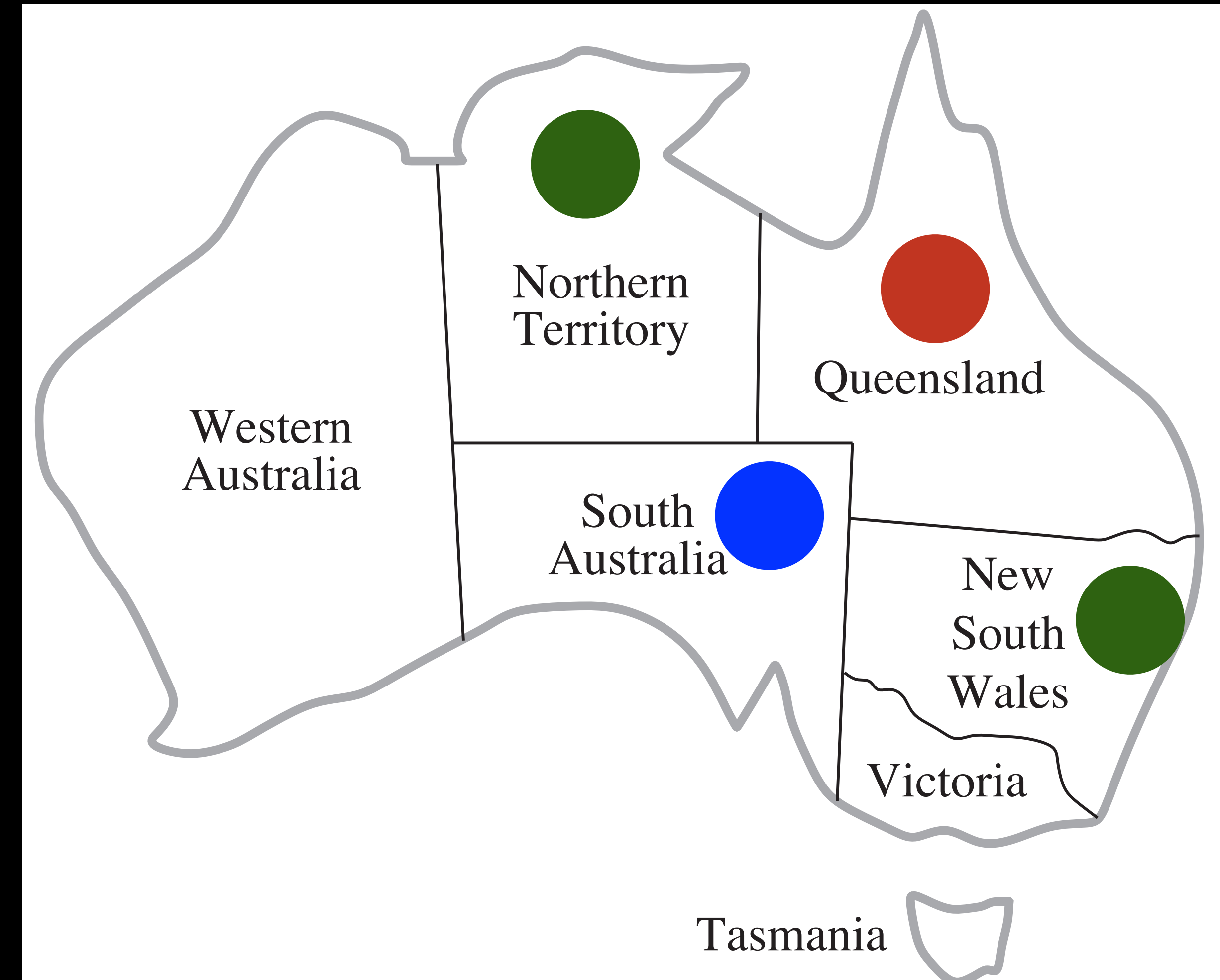


NT and NSW are forced G

Remaining possibilities:

$2^2 \times 3 = 12$

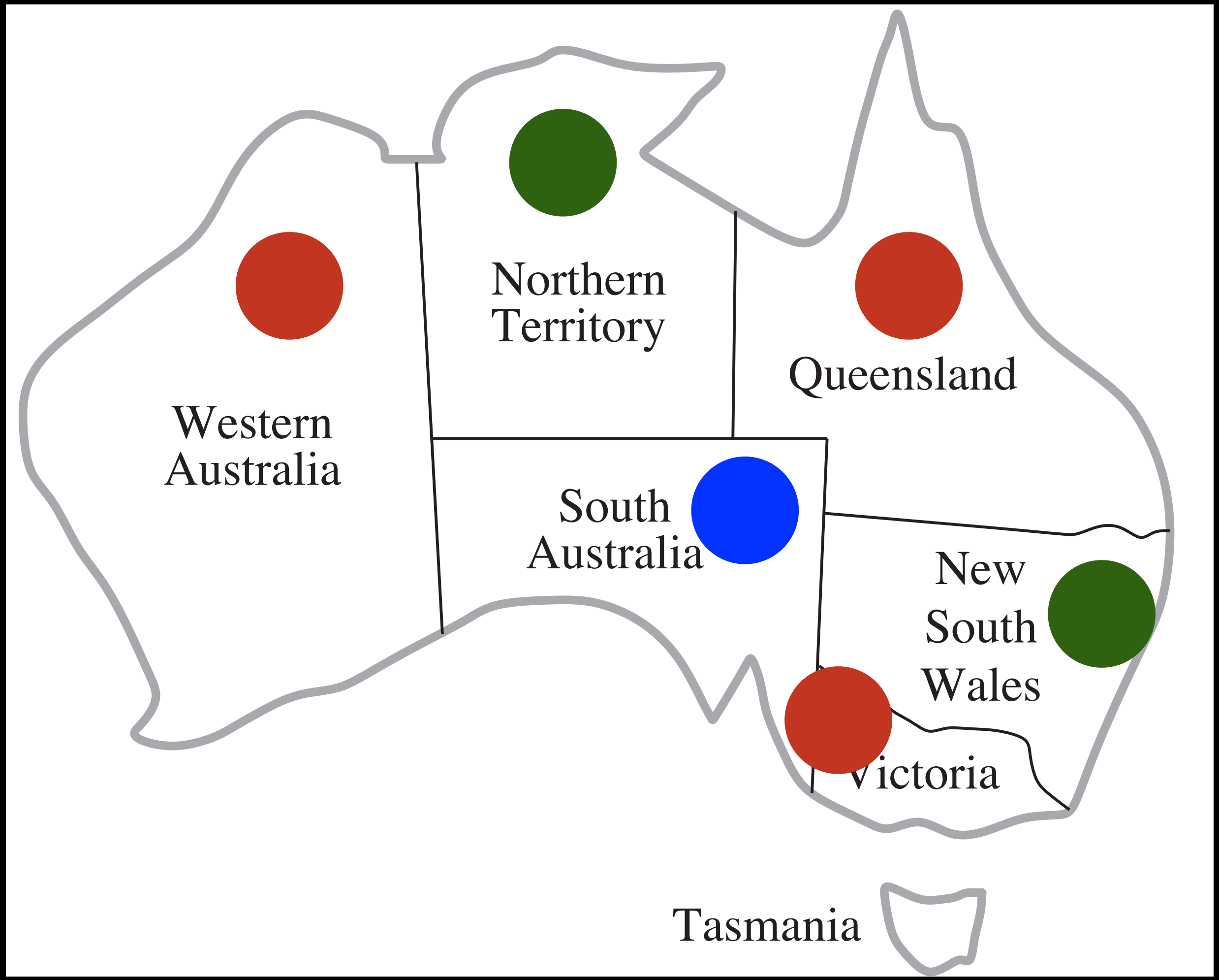
WA	R
NT	G
SA	B
Q	R
NSW	G
V	R
T	R, G, B



Simplify: remove G from adjacent regions

Remaining possibilities:
3

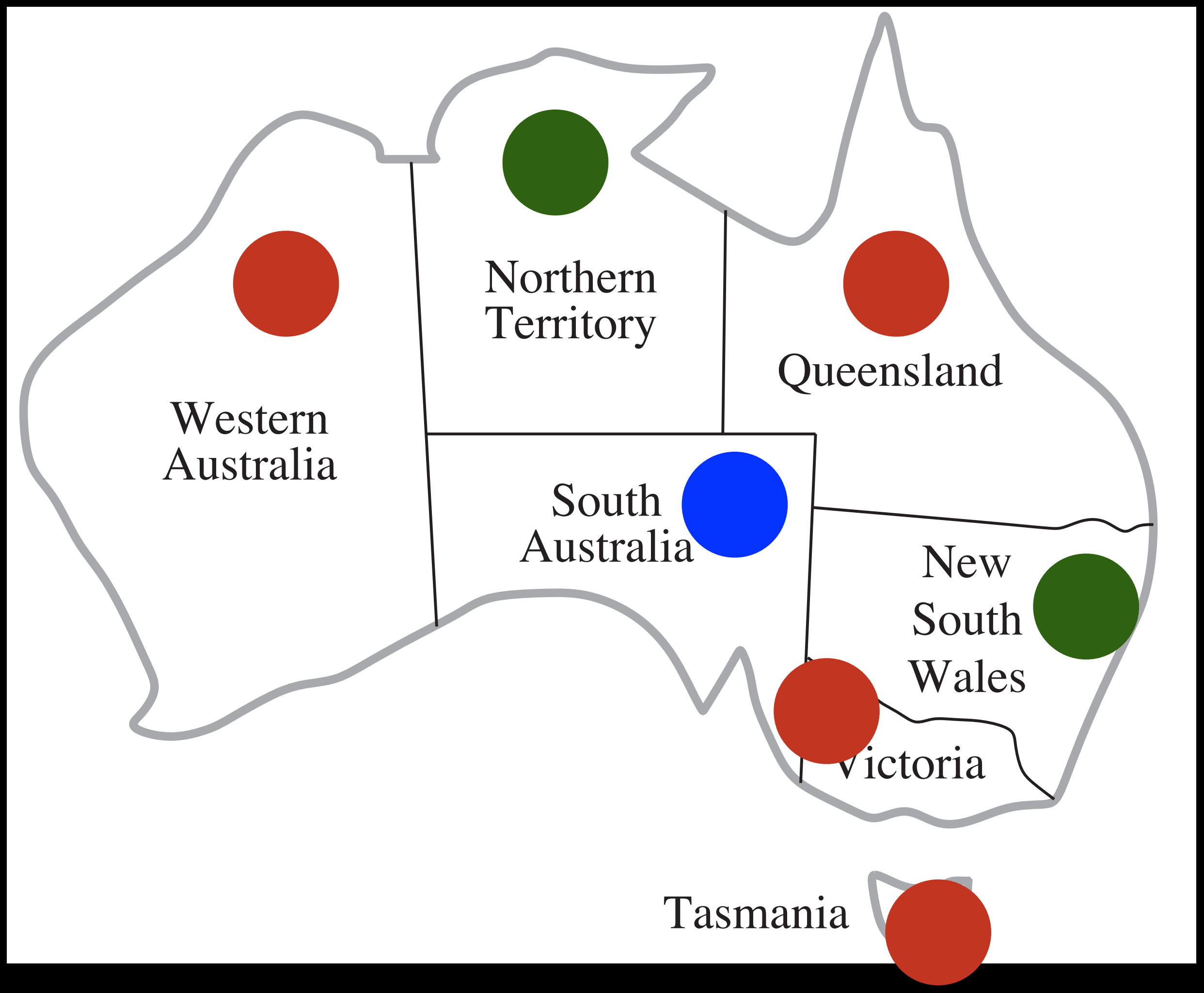
WA	R
NT	G
SA	B
Q	R
NSW	G
V	R
T	R, G, B



WA and V are forced red

Remaining possibilities:
3

WA	R
NT	G
SA	B
Q	R
NSW	G
V	R
T	R



Choose: any color for T

Solved!

Constraint Propagation

- Using the constraints to reduce the set of legal values of a variable, which can in turn reduce the legal values of another variable, and so on
- Not a search process itself!
  - Part of state update in state-space search
- A type of inference: making implicit information explicit

Arc-Consistency

- The particular kind of constraint propagation we just saw is called arc-consistency
  - Why? Because it involves considering 2 nodes at a time (the ends of an arc)
- There are other kinds of constraint propagation, but arc-consistency is usually the most practical

Constraint Propagation

- Can be used as pre-processing step for any kind of search
  - Including local search
- Can be interleaved with any kind of search over partial assignments, where the action is "assign a value to an unassigned variable"
  - Popular choice: depth-first search

Domain-Independent Heuristics

- There are good heuristics for deciding which variable to assign next
- Choose one with the smallest domain
    - Maximizes likelihood of making a correct choice!
- Choose one involved in largest number of constraints
    - Likely to lead to lots of constraint propagation!

Check Your Understanding

- Why can't you use constraint propagation after each step of local search?

Check Your Understanding

- Why can't you use constraint propagation after each step of local search?
- Because local search is over complete states
  - Every variable has a particular value
  - You can't therefore remove a value from the domain of a variable

CSPs Summary

- Impose a structure on the representation of states: Variables, Domains, Constraints
- Backtracking search for complete, consistent assignment of values to variables
- Inference (constraint propagation) can reduce the domains of variables
  - Preprocessing or interleaved with search