Local Search
Constraint Satisfaction Problems

Russell and Norvig: Chapter 4
CSE 240: Winter 2023
Lecture 9
Guest Lecturer: Prof. Razvan Marinescu

Announcements

- This week: Prof. Marinescu will lecture (Prof. Gilpin at AAAI)
- Prof. Gilpin will *briefly* go over Assignment 3 (posted on Canvas with instructions).

Agenda and Topics

- Assignment 3 Overview
- Local search and optimization algorithms.
    - Genetic Algorithms
    - Local search in continuous space
- Constraint Satisfaction Problems

Assignment 3

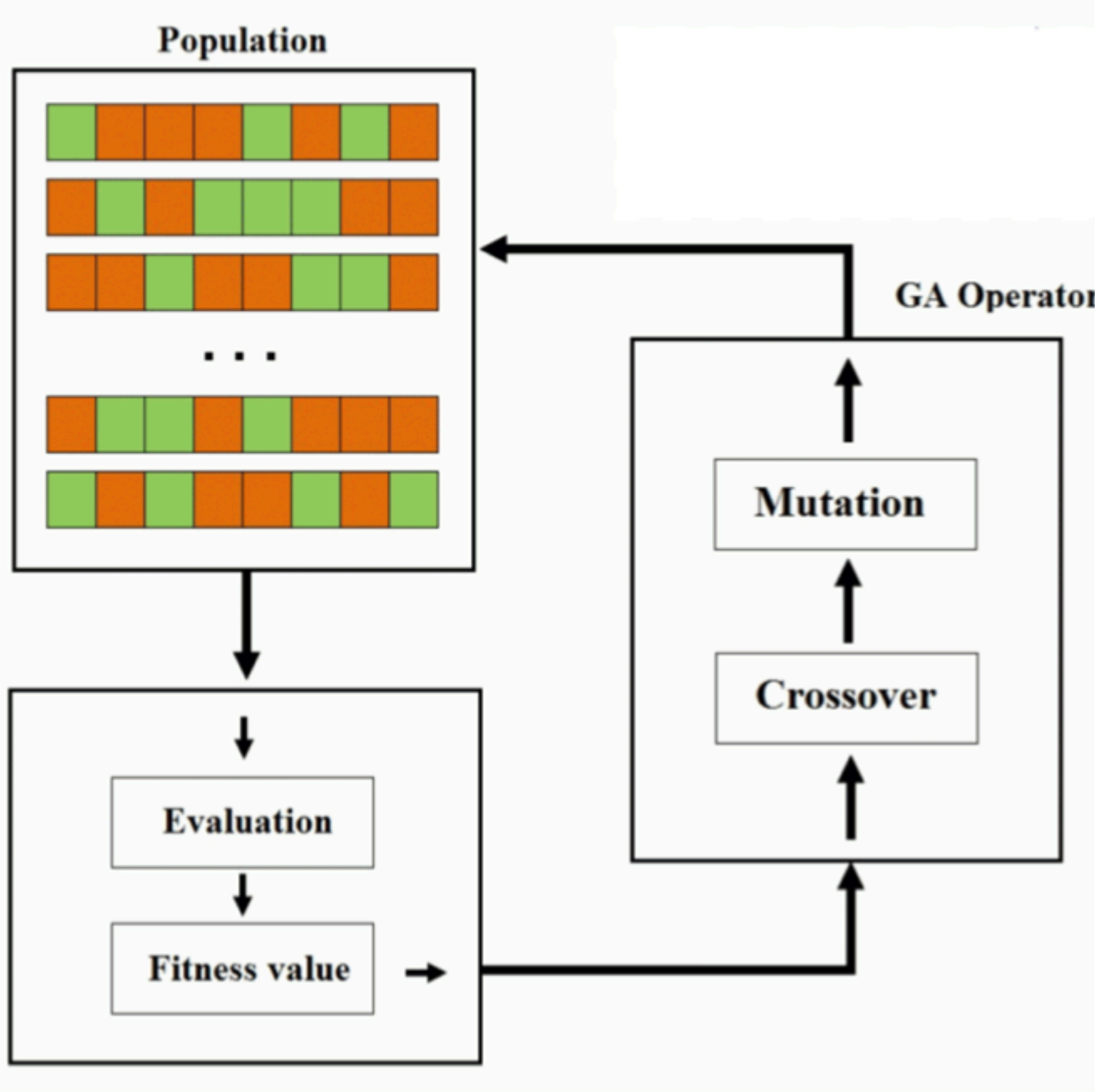
Genetic Algorithms

Genetic Algorithms

- Quicker but randomized searching for an optimal parameter vector
- Operations
  - Crossover (2 parents -> 2 children)
  - Mutation (one bit)
- Basic structure
  - Create population
  - Perform crossover & mutation (on the fittest)
  - Keep only fittest children

Genetic Algorithms

- State = a string over a finite alphabet (an individual)
  - A successor state is generated by combining two parent states
- Start with k randomly generated states (population)
- Evaluation function (fitness function): Higher values for better states.
- Select individuals for next generation based on fitness
  - P(indiv. in next gen) = indiv. fitness / total population fitness
- Crossover: fit parents to yield next generation (offspring)
- Mutate the offspring randomly with some low probability



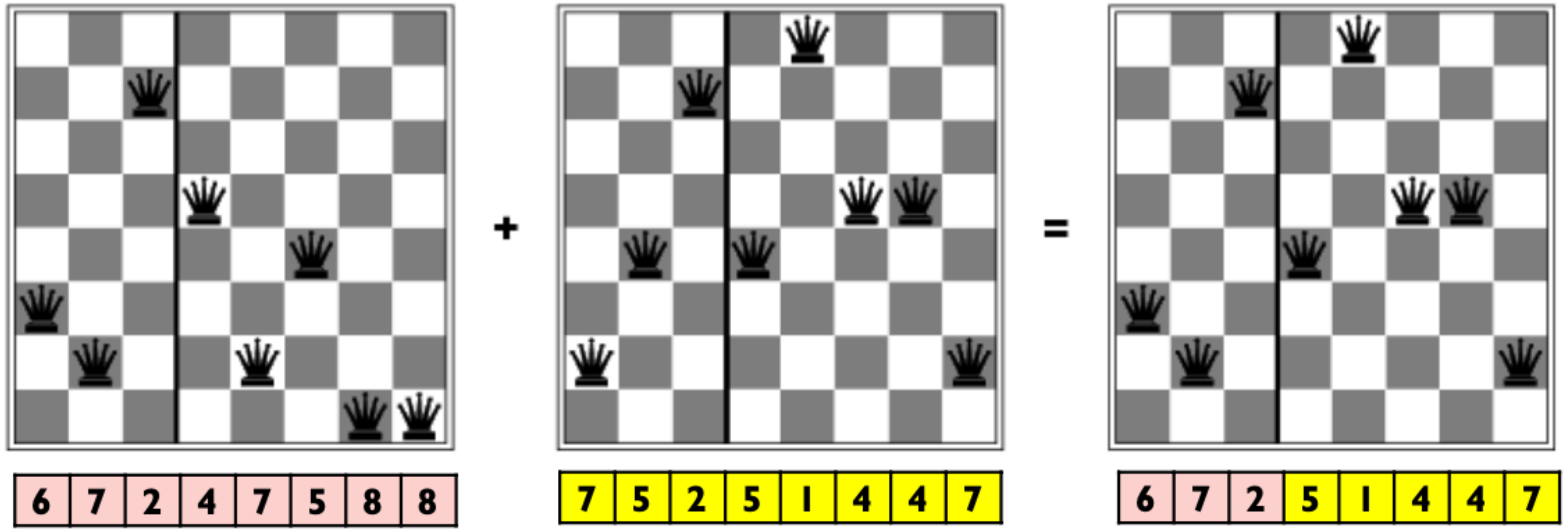
Describe an Individual

- Describe the individual (or state) as a string
- Fitness function: number of non-attacking pairs of queens
    - 24 in this example.

2	4	7	4	8	5	5	2

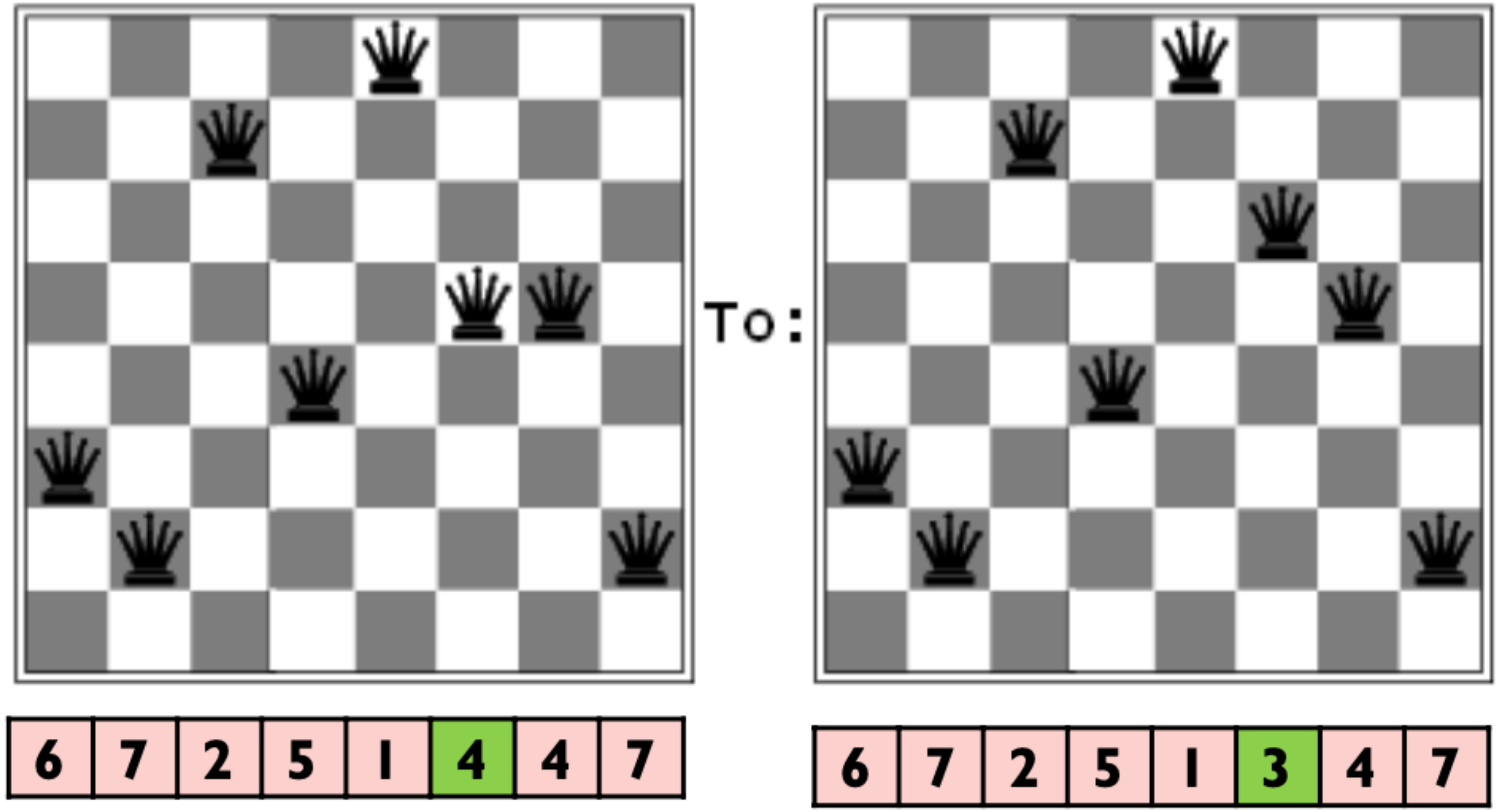
Crossover

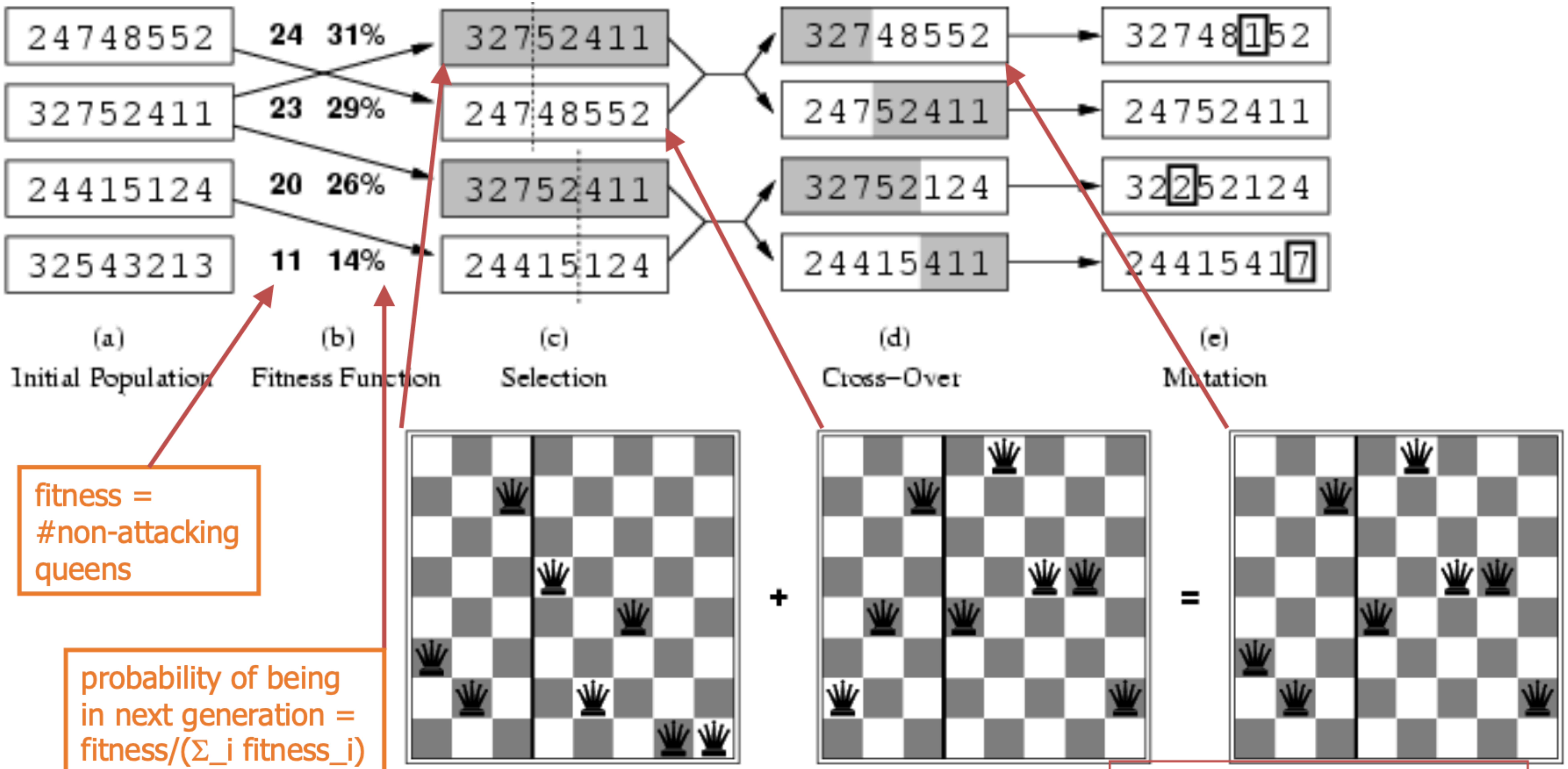
To select some part of the state from one parent and the rest from another



Mutation

To change a small part of one state with a small probability





fitness = #non-attacking queens

probability of being in next generation = fitness/(Σ_i fitness_i)

How to convert a fitness value into a probability of being in the next generation.

- Fitness function: #non-attacking queen pairs
  - min = 0, max = 8 × 7/2 = 28
- Σ_i fitness_i = 24+23+20+11 = 78
- P(pick child_1 for next gen.) = fitness_1/(Σ_i fitness_i) = 24/78 = 31%
- P(pick child_2 for next gen.) = fitness_2/(Σ_i fitness_i) = 23/78 = 29%; etc

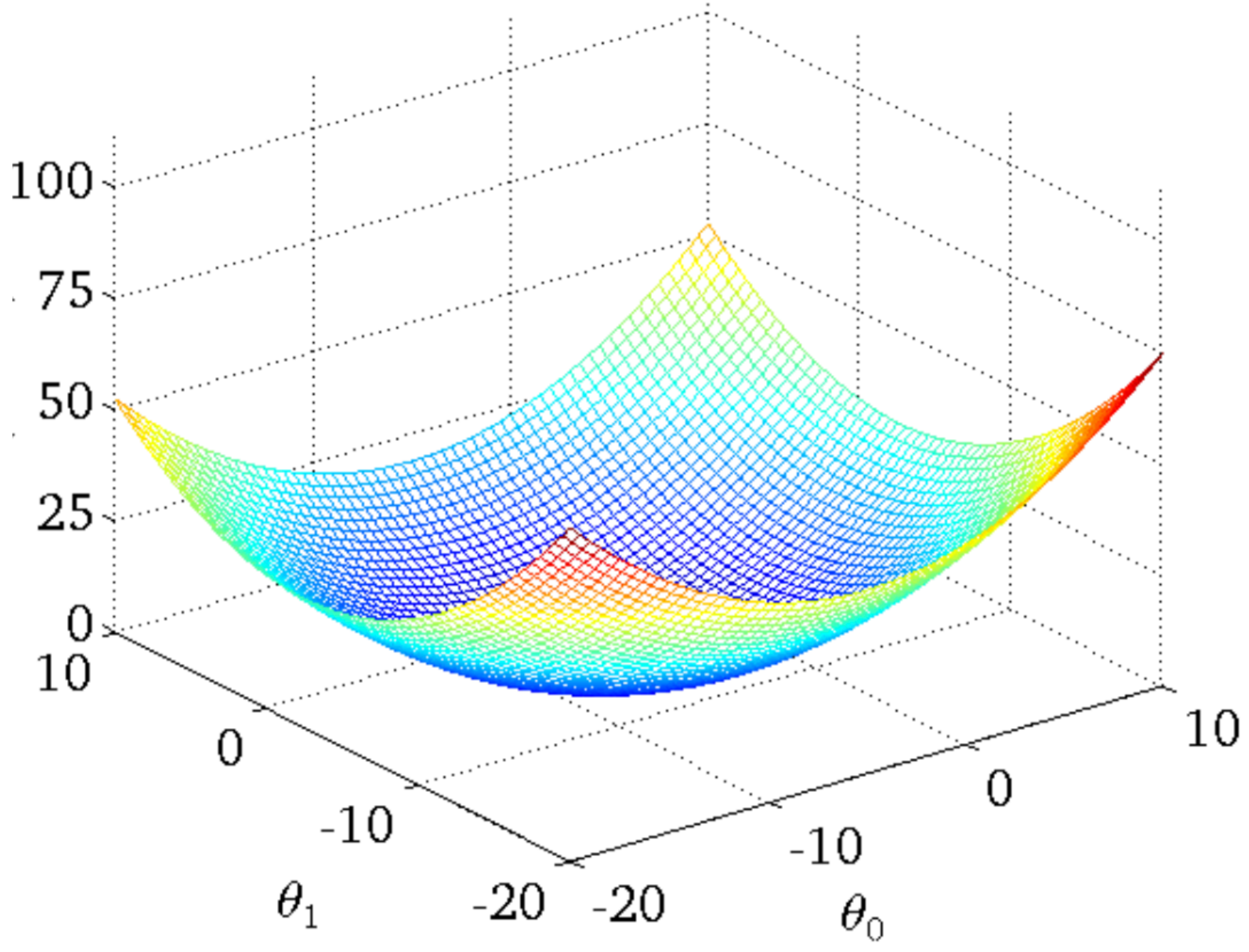
Comments On Genetic Algorithm

- Genetic algorithm is a variant of “stochastic beam search”
- Positive points:
  - Random exploration can find solutions that local search cannot
  - Appealing connection to human evolution
- Negative points:
  - Large number of tunable parameters
  - Not convincing that the algorithm performs better than hill-climbing with random restart in general

Local Search in Continuous Space

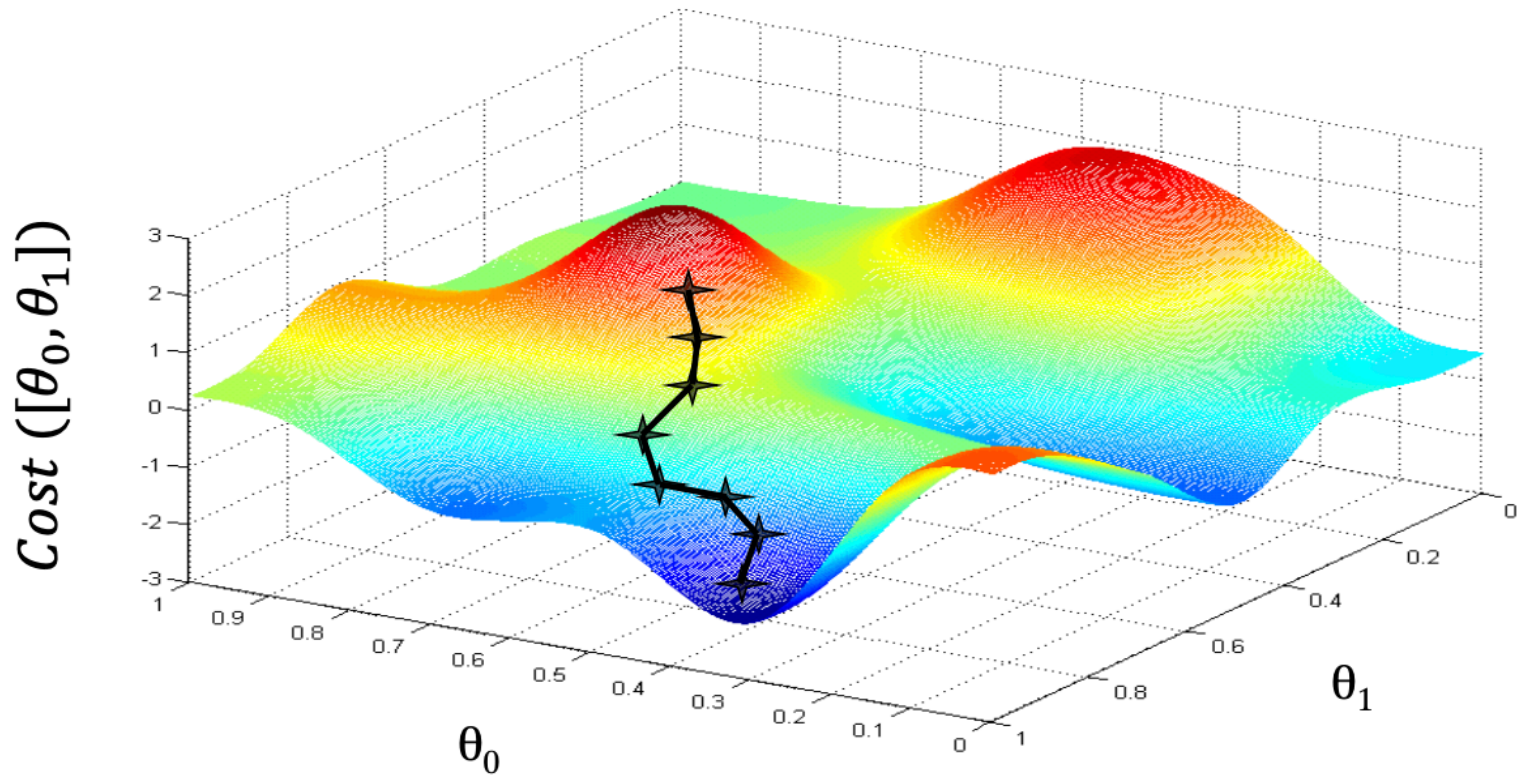
Example of a Continuous Landscape

$$h([\theta_0, \theta_1]) = Cost([\theta_0, \theta_1])$$



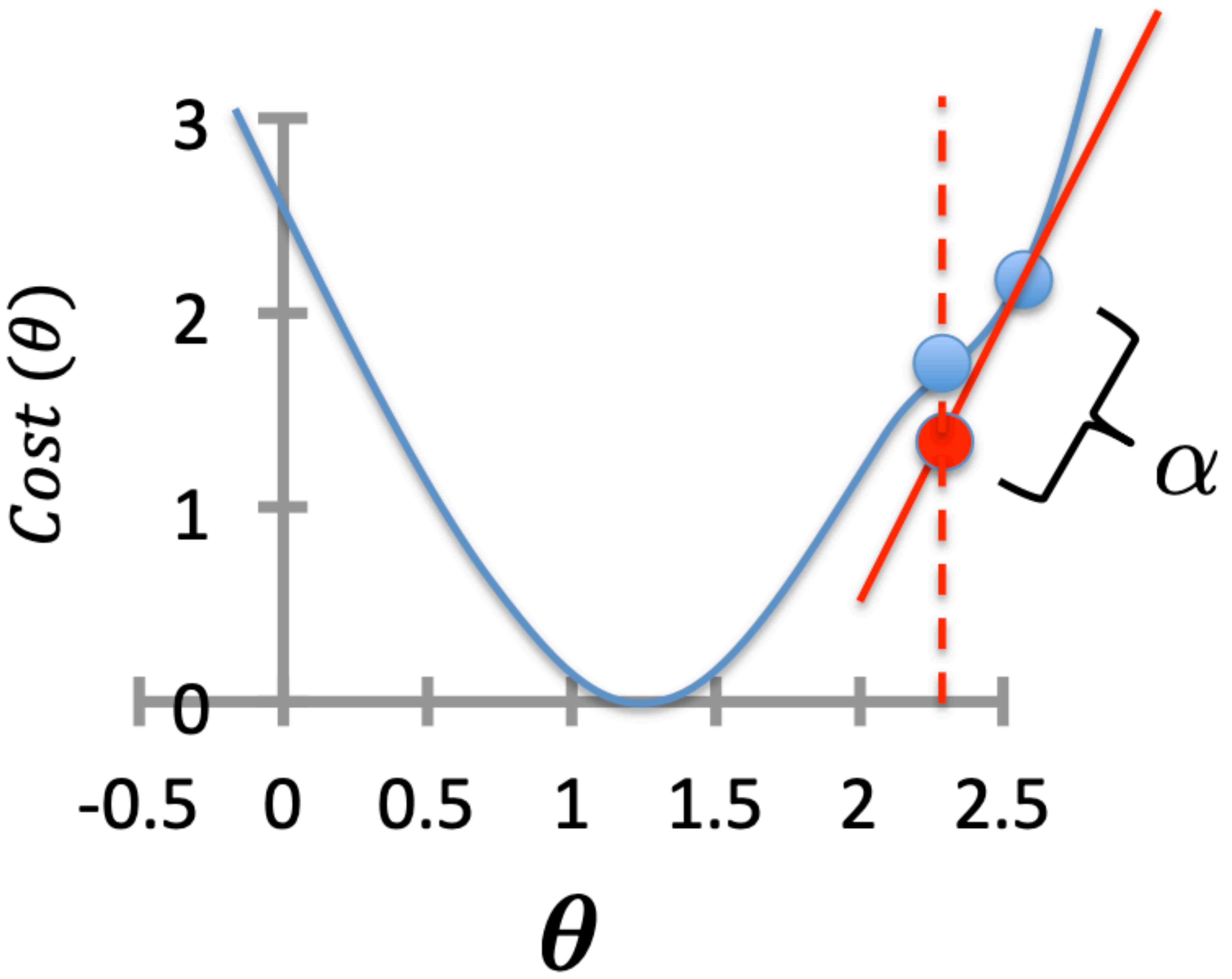
Gradient Descent

- Choose an initial state: $\theta = [\theta_0, \theta_1]$ (for this 2 dimensional example).
- Until we reach a global/local minimum:
  - Move to a better successor state
  - Choose a new value of θ to reduce $Cost(\theta)$



Gradient Descent Algorithm

- Initialize θ
- Repeat until convergence:
  - $\theta \leftarrow \theta - \alpha \dfrac{\partial Cost(\theta)}{\partial \theta}$
  - α is the learning rate.



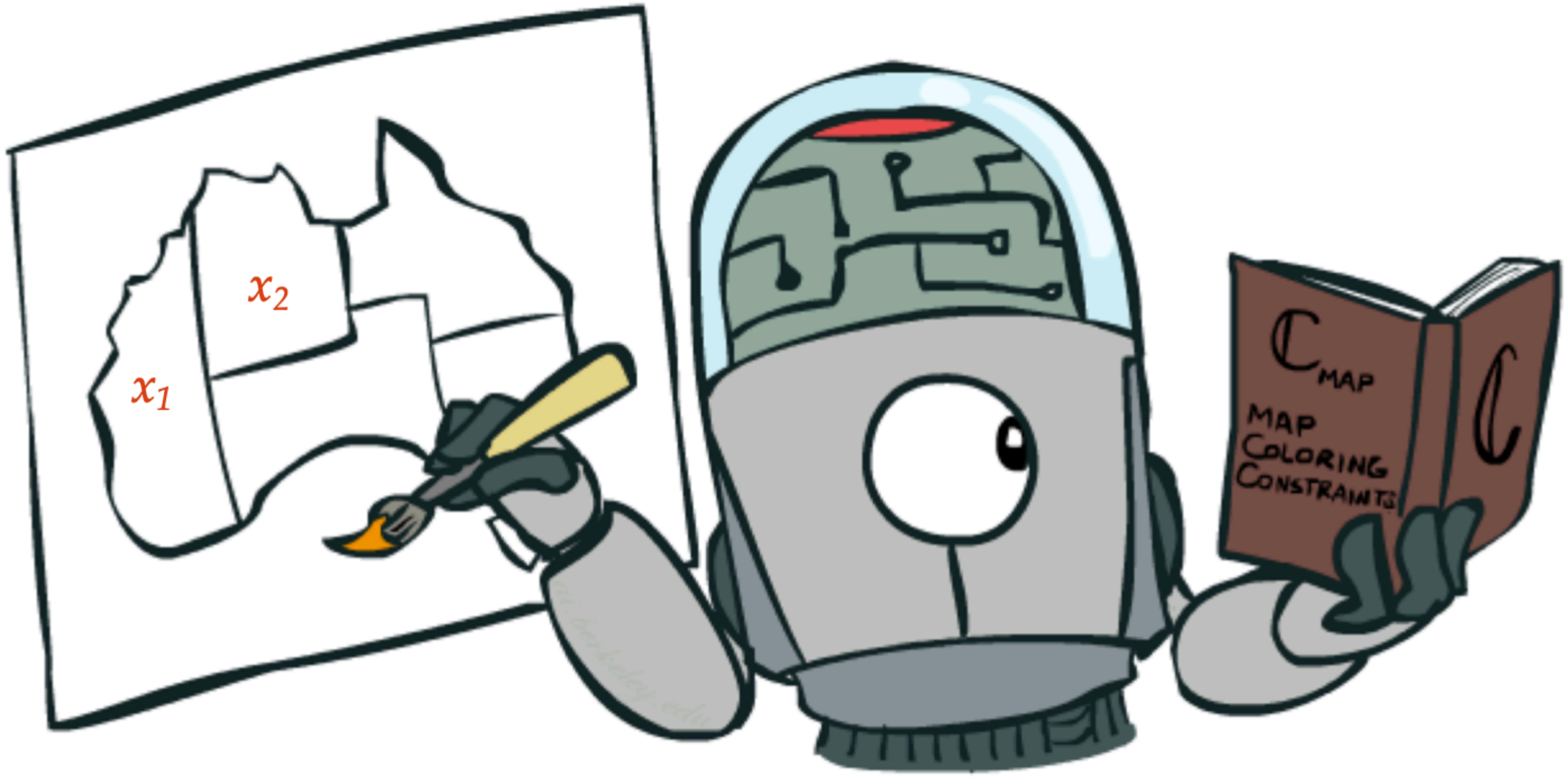
Constraint Satisfaction Problems

Constraint Satisfaction Problems

N variables

domain D

constraints



states

partial assignment

goal test

complete; satisfies constraints

successor function

assign an unassigned variable

What is Search For?

- Assumptions about the world: a single agent, deterministic actions, fully-observed state, discrete state space
- Planning: sequences of actions
  - The path to the goal is the important thing
  - Paths have various costs
  - Heuristics give problem-specific guidance
- Identification: assignments to variables
  - The goal itself is important, not the path
  - All goals at the same depth (for some formulations)
  - CSPs are specialized for identification problems

Exercise: Sudoku

- But, before we get into all that, let's do a puzzle…

- http://www.sudokuwiki.org/sudoku.htm

- Each Sudoku has a unique solution that can be reached logically without guessing. Enter digits from 1 to 9 into the blank spaces. Every row must contain one of each digit. So must every column, as must every 3x3 square.

Take 5 minutes

CE 9: Sudoku Puzzle

	1	2	3	4	5	6	7	8	9
1	7		5	1	6	4		2	8
2	4		6		7			1	5
3	1				3		6	7	4
4	2	4	9	3	8	1	7	5	6
5	3	8	7	2	5		1	4	9
6	5	6	1	7	4	9	8	3	2
7	8	5	2	6	1	7	4	9	3
8	9	1	4	8	2	3	5	6	7
9	6	7	3	4	9	5	2	8	1

Problem Formulation

- What is the state space?
- What is the initial state?
- What is the successor function?
- What is the goal test?

CSP as a Search Problem

- Initial state: empty assignment
- Successor function: a value is assigned to any unassigned variable
- Goal test: the assignment is complete and consistent

We can do better…….

What else is needed?

- Not just a successor function and goal test
- But also a means to propagate the constraints imposed by one move on the others and an early failure test
- → Explicit representation of constraints and constraint manipulation algorithms

What is Search For?

- **Planning:** sequences of actions
  - The path to the goal is the important thing
  - Paths have various costs, depths
  - Heuristics to guide, fringe to keep backups

- **Identification:** assignments to variables
  - The goal itself is important, not the path
  - All paths at the same depth (for some formulations)
  - CSPs are specialized for identification problems

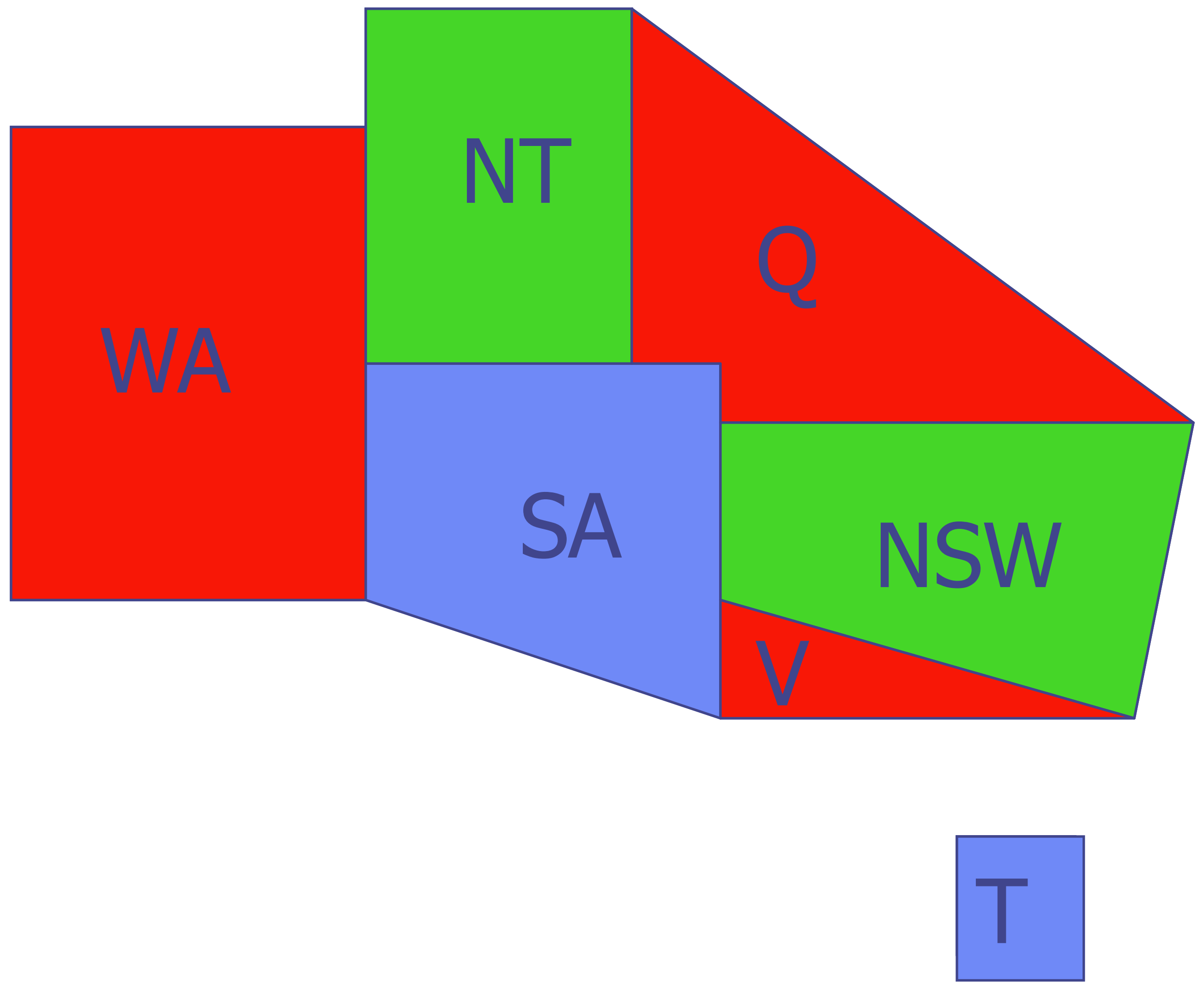
Constraint Satisfaction Problem

- Set of variables $\{X_1, X_2, \ldots, X_n\}$
- Each variable X_i has a domain D_i of possible values
  - Usually D_i is discrete and finite
- Set of constraints $\{C_1, C_2, \ldots, C_p\}$
  - Each constraint C_k involves a subset of variables and specifies the allowable combinations of values of these variables
- Goal: Assign a value to every variable such that all constraints are satisfied

Example: Sudoku CSP

- variables: $X_{11}, \ldots, X_{99}$
- domains: $\{1,\ldots,9\}$
- constraints:
  - row constraint: $X_{11} \neq X_{12}, \ldots, X_{11} \neq X_{19}$
  - col constraint: $X_{11} \neq X_{21}, \ldots, X_{11} \neq X_{91}$
  - block constraint: $X_{11} \neq X_{12}, \ldots, X_{11} \neq X_{33}$
- Goal: Assign a value to every variable such that all constraints are satisfied

Example: Map Coloring



- 7 variables {WA,NT,SA,Q,NSW,V,T}
- Each variable has the same domain {red, green, blue}
- No two adjacent variables have the same value:

WA≠NT, WA≠SA, NT≠SA, NT≠Q, SA≠Q, SA≠NSW, SA≠V,Q≠NSW, NSW≠V

Example: Map Coloring

- Variables: $\mathrm{WA},\ \mathrm{NT},\ \mathrm{Q},\ \mathrm{NSW},\ \mathrm{V},\ \mathrm{SA},\ \mathrm{T}$

- Domains: $D = \{\mathrm{red}, \mathrm{green}, \mathrm{blue}\}$

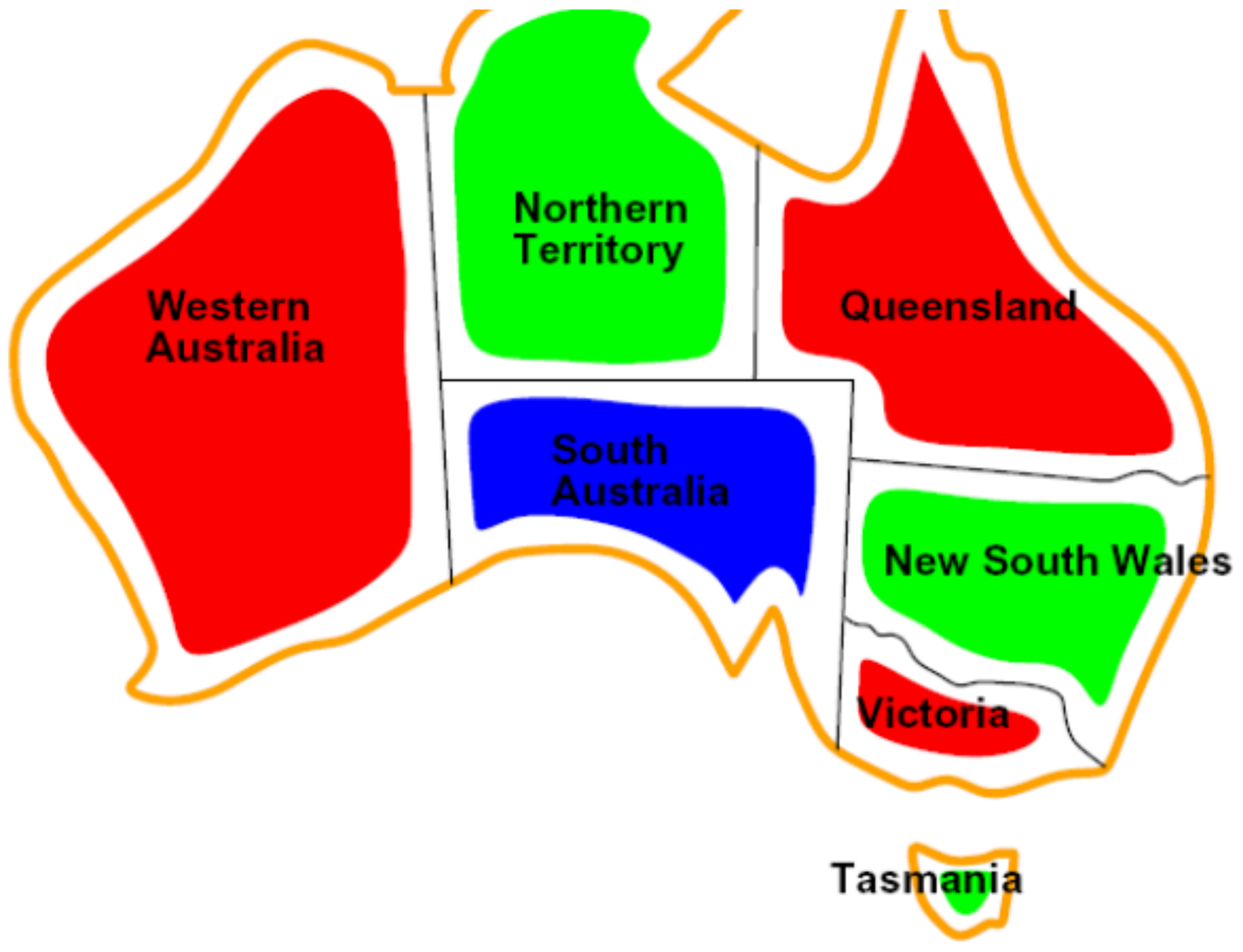
- Constraints: adjacent regions must have different colors

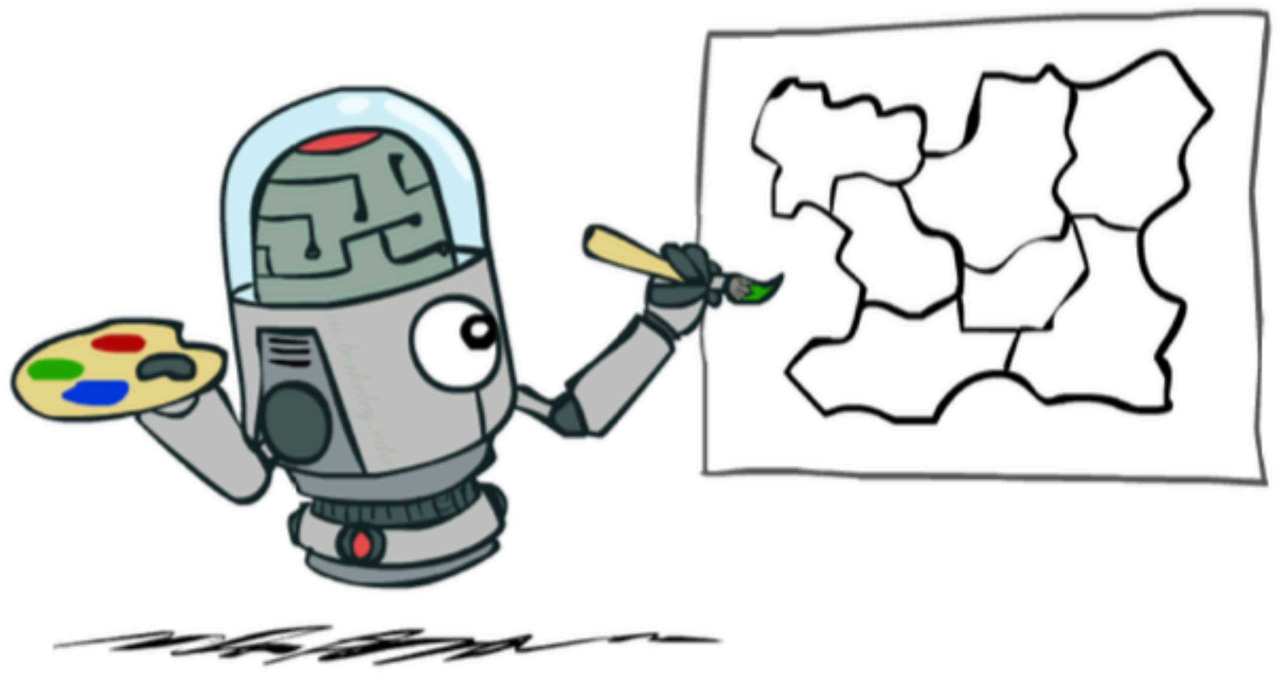
  Implicit: $\mathrm{WA} \neq \mathrm{NT}$

  Explicit: $(\mathrm{WA}, \mathrm{NT}) \in \{(\mathrm{red}, \mathrm{green}), (\mathrm{red}, \mathrm{blue}), \ldots\}$

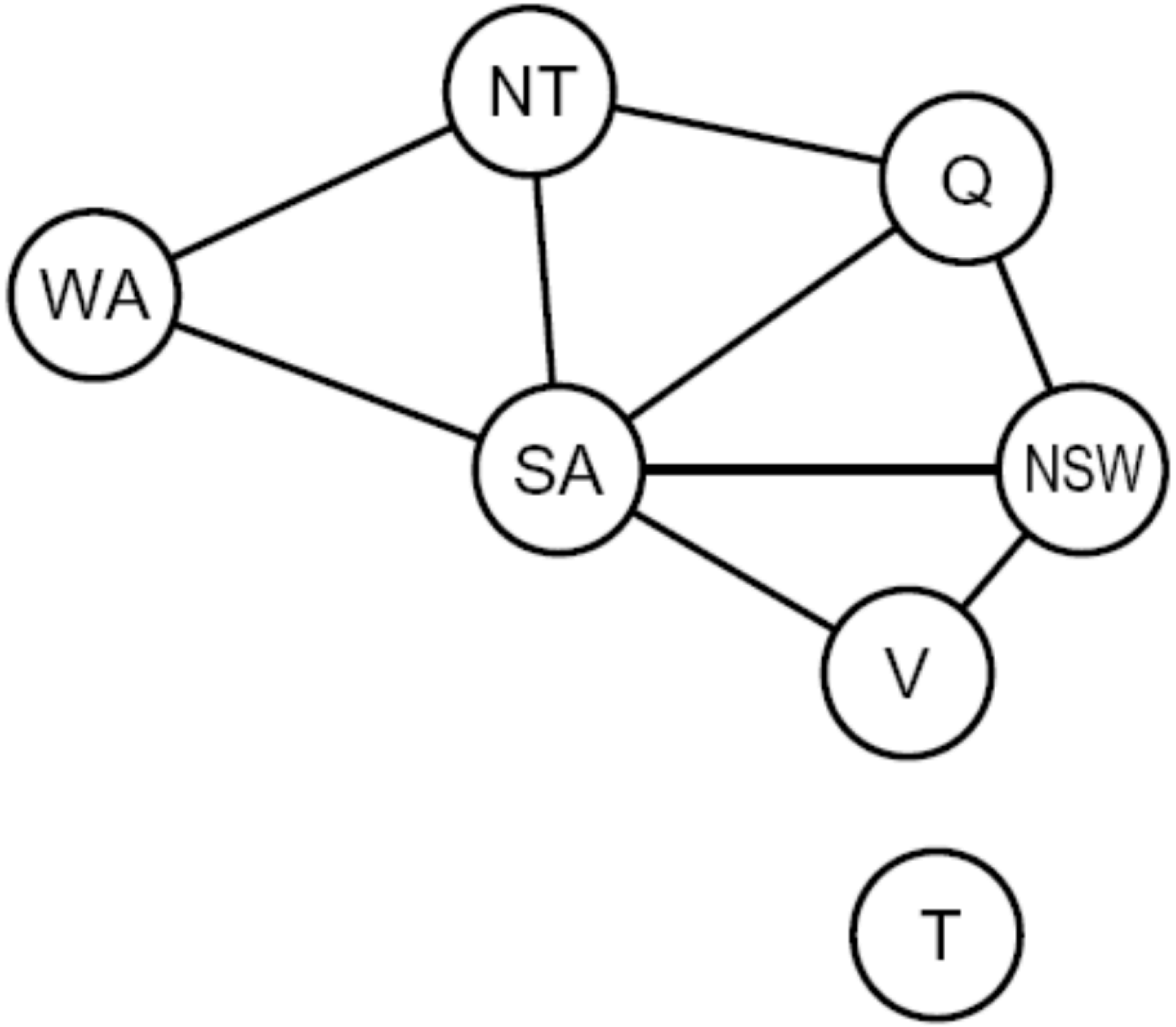
- Solutions are assignments satisfying all constraints, e.g.:

  $\{\mathrm{WA}{=}\mathrm{red},\ \mathrm{NT}{=}\mathrm{green},\ \mathrm{Q}{=}\mathrm{red},\ \mathrm{NSW}{=}\mathrm{green},\ \mathrm{V}{=}\mathrm{red},\ \mathrm{SA}{=}\mathrm{blue},\ \mathrm{T}{=}\mathrm{green}\}$



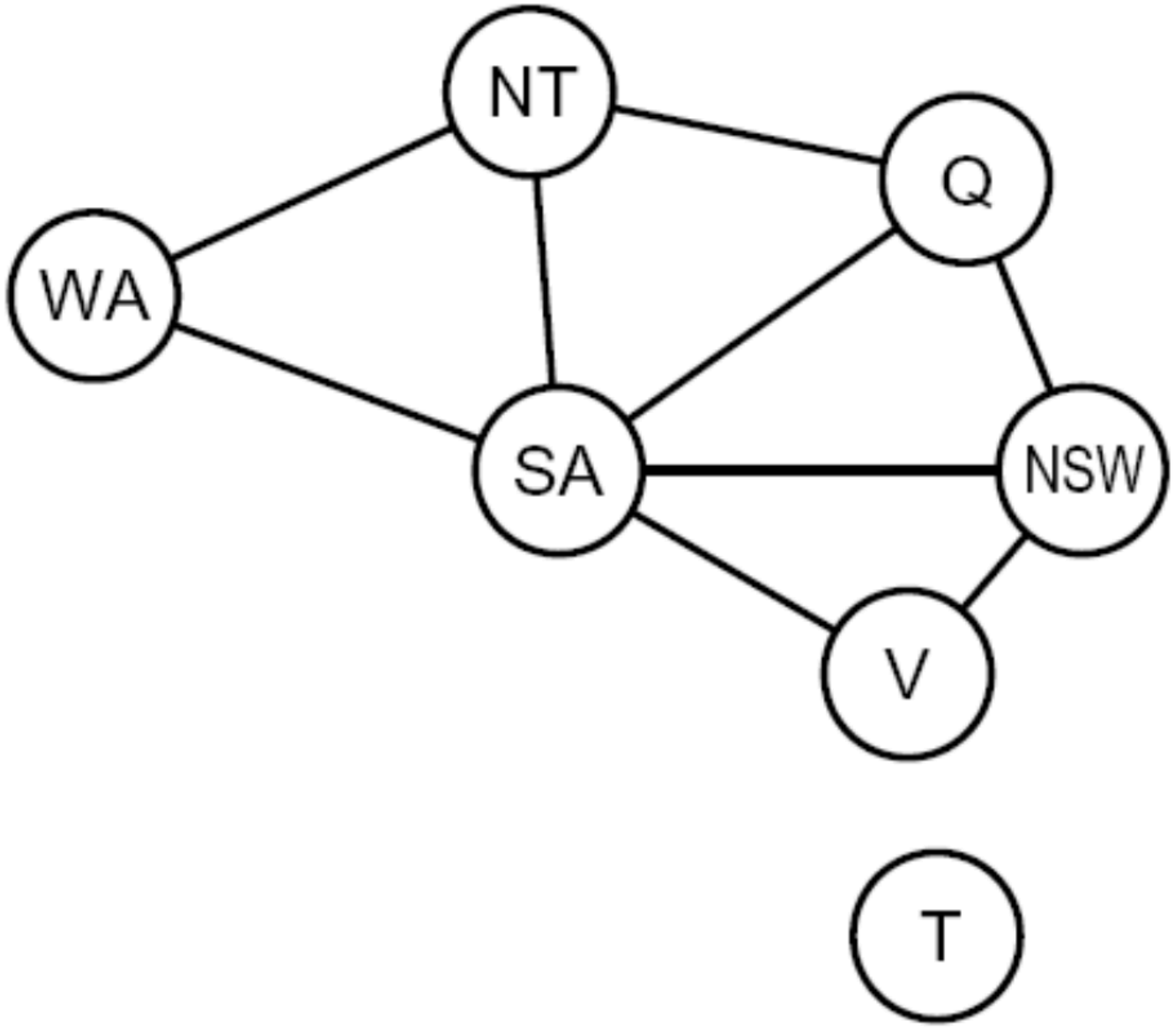


Constraint Graphs

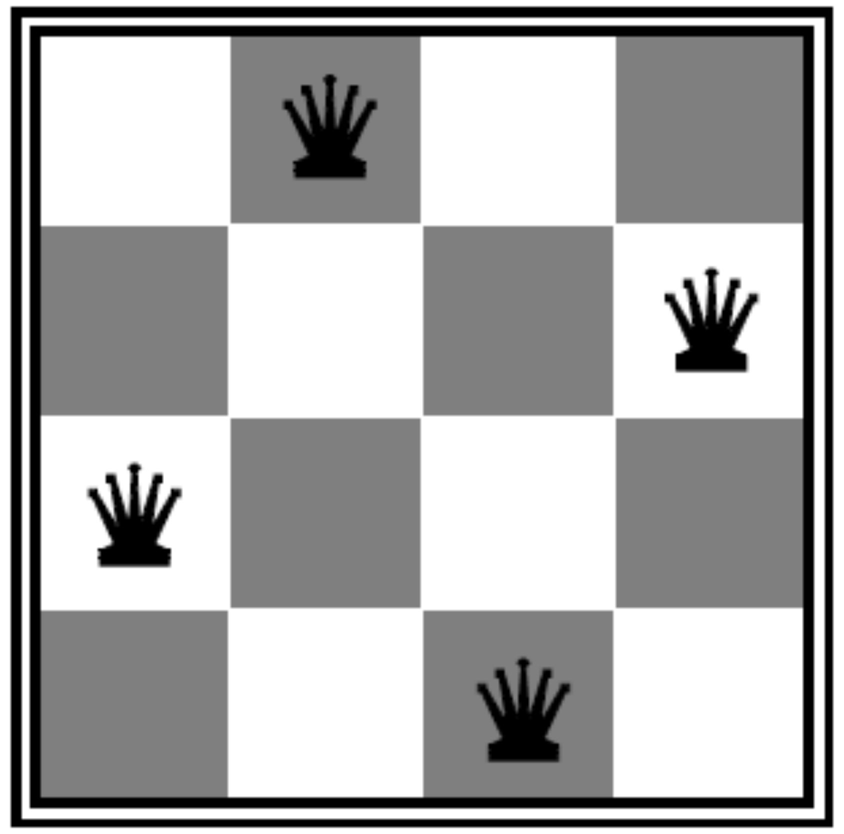


Constraint Graphs

- Binary CSP: each constraint relates (at most) two variables
- Binary 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 sub-problem!



Example: N-Queens

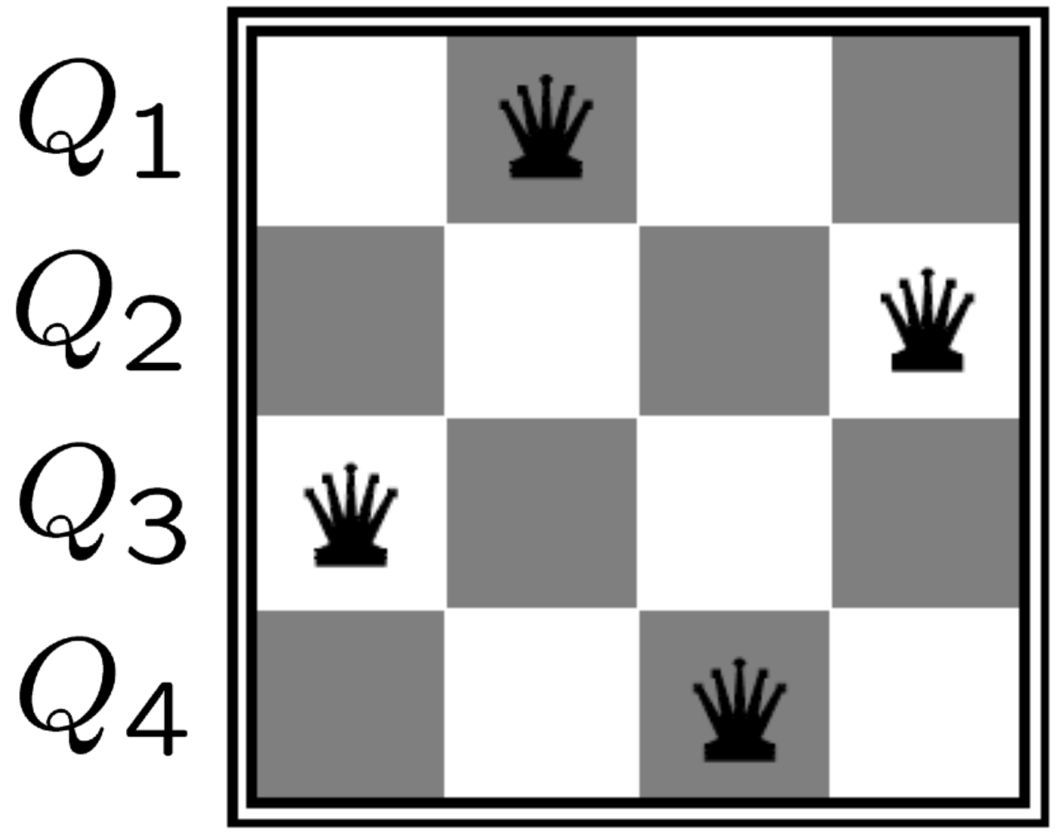


- Formulation 1:
  - Variables: X_{ij}
  - Domains: $\{0, 1\}$
  - Constraints

$$\forall i, j, k \;\; (X_{ij}, X_{ik}) \in \{(0,0), (0,1), (1,0)\}$$
$$\forall i, j, k \;\; (X_{ij}, X_{kj}) \in \{(0,0), (0,1), (1,0)\}$$
$$\forall i, j, k \;\; (X_{ij}, X_{i+k,j+k}) \in \{(0,0), (0,1), (1,0)\}$$
$$\forall i, j, k \;\; (X_{ij}, X_{i+k,j-k}) \in \{(0,0), (0,1), (1,0)\}$$

$$\sum_{i,j} X_{ij} = N$$

Example: N-Queens



- Formulation 2:
  - Variables: Q_k
  - Domains: $\{1, 2, 3, \ldots N\}$
  - Constraints:

Implicit: $\forall i, j \;\; \text{non-threatening}(Q_i, Q_j)$

-or-

Explicit: $(Q_1, Q_2) \in \{(1,3), (1,4), \ldots\}$

$\cdots$

Example: Cryptarithmetic

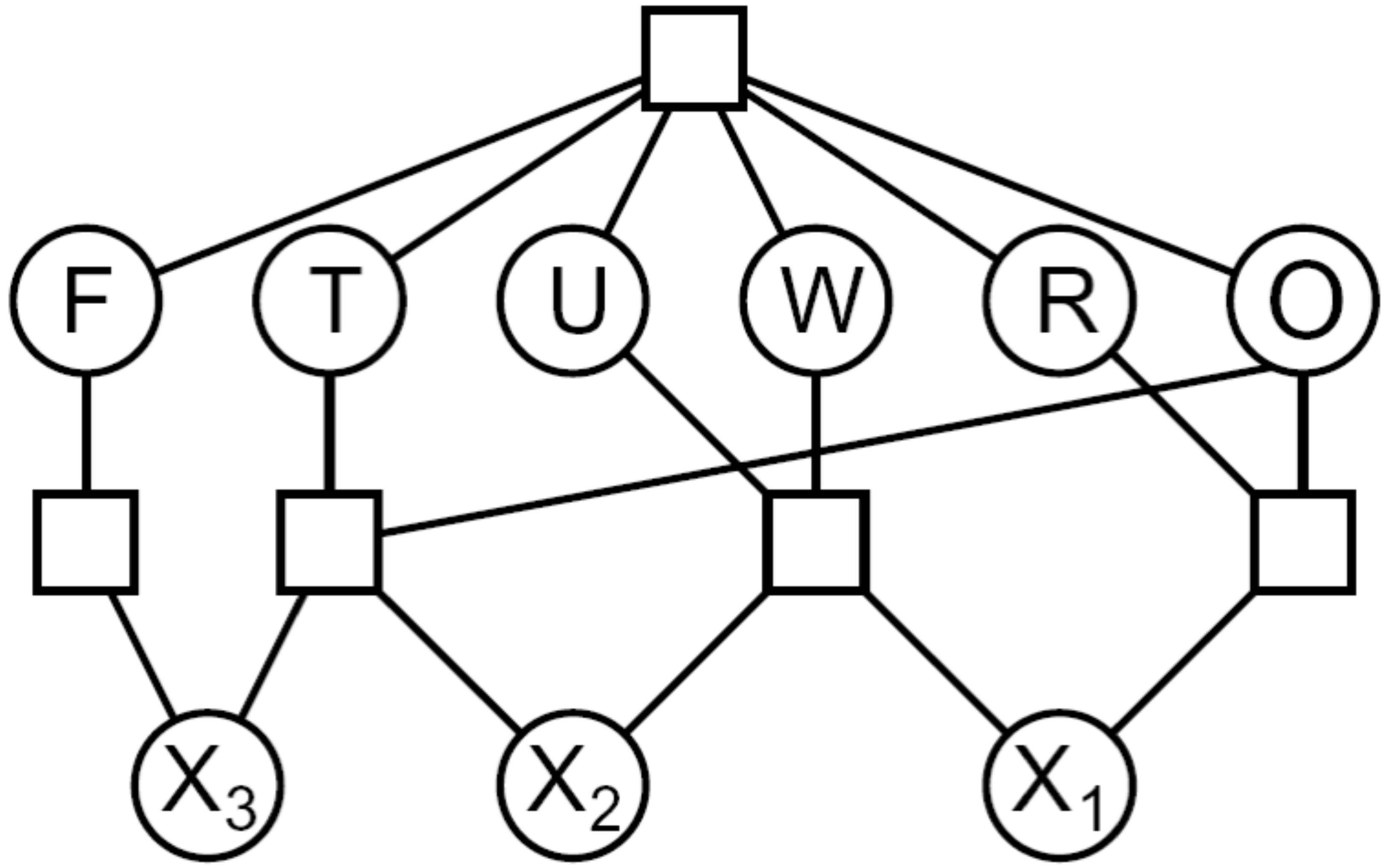
- Variables: $F \ T \ U \ W \ R \ O \ X_1 \ X_2 \ X_3$
- Domains: $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- Constraints:

$$\text{alldiff}(F, T, U, W, R, O)$$

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

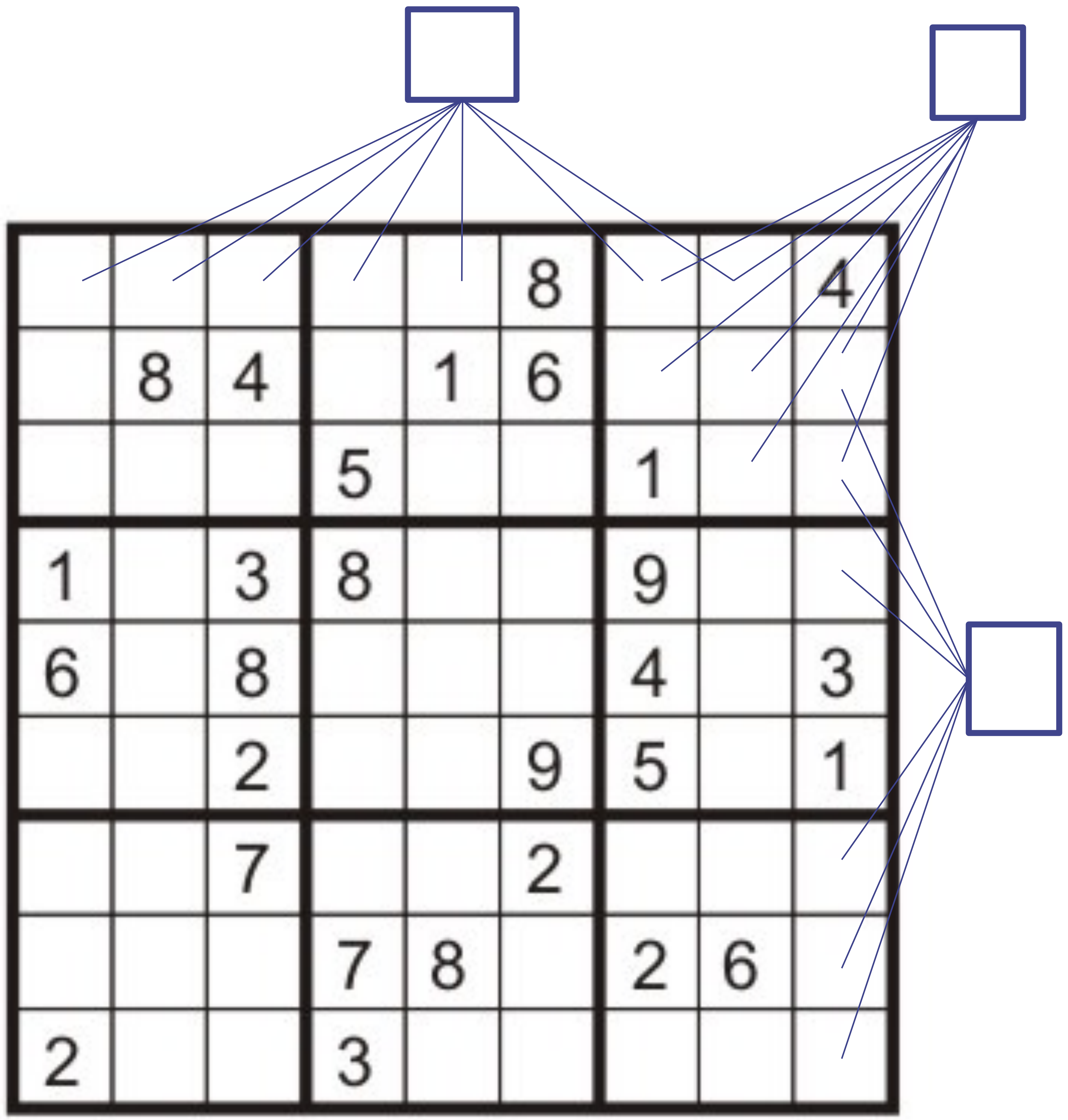
$$\cdots$$

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



Aside, solution: O=7, R=4, W=6, U=2, T=8, F=1; 867 + 867 = 1734

Sudoku Constraint Graph



- Variables:
  - Each (open) square
- Domains:
  - {1,2,…,9}
- Constraints:

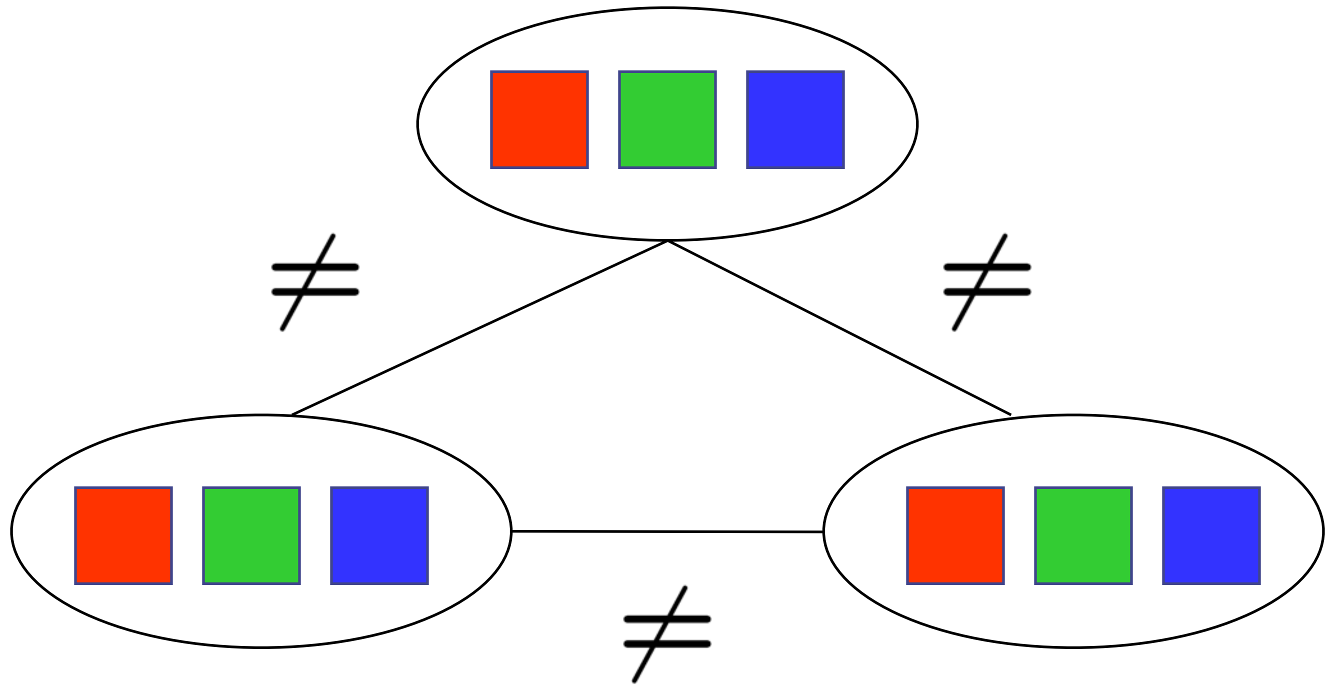
9-way alldiff for each column

9-way alldiff for each row

9-way alldiff for each region

CSP Problem Formulation

- CSPs:
  - Variables
  - Domains
  - Constraints
    - Implicit (provide code to compute)
    - Explicit (provide a subset of the possible tuples)



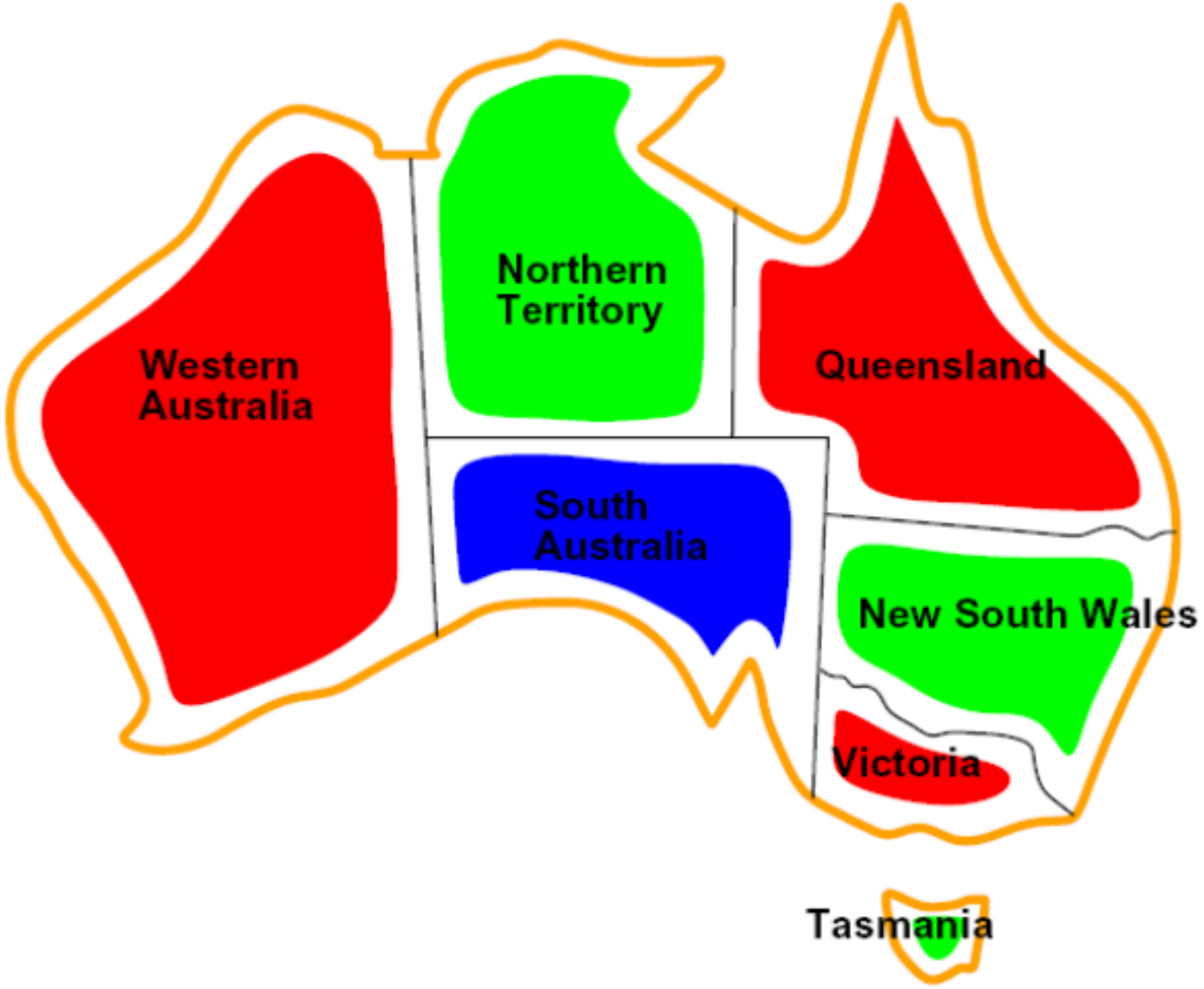
- Unary Constraints
- Binary Constraints
- N-ary Constraints

Standard Search Formulation

- Standard search formulation of CSPs
- States defined by the values assigned so far (partial assignments)
  - Initial state: the empty assignment, {}
  - Successor function (Actions): assign a value to an unassigned variable
  - Goal test: the current assignment is complete (all variables have assigned values) and consistent (satisfies all constraints)
  - Path cost: not important
- We’ll start with the straightforward, naïve approach, then improve it

Search Methods

- What would DFS do?
- What would BFS do?
- What problems does naïve search have?



Summary and Next Time

- Assignment 3 Overview
- Local search and optimization algorithms.
  - Genetic Algorithms
  - Local search in continuous space
- Constraint Satisfaction Problems

- Constraint Satisfaction Problems
  - Algorithms
    - Backtracking
    - Arc Consistency
    - Path Consistency