# Intelligent Agents

## Constraint Satisfaction Problems

Curtis Larsen

Utah Tech University—Computing

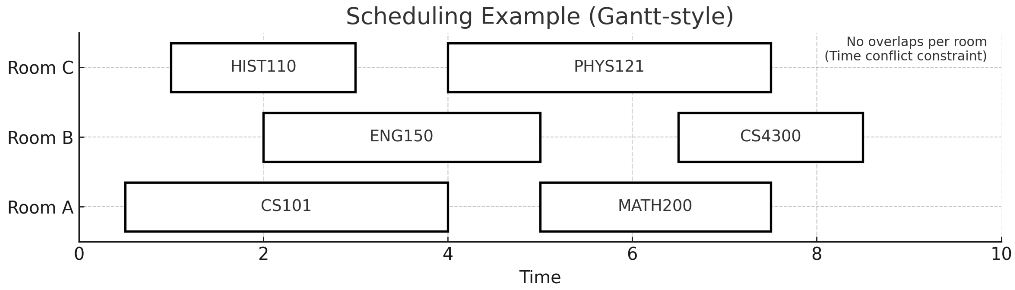
Fall 2025

# Motivation: Why CSPs?

- Many real-world problems can be naturally expressed as **variables with domains** subject to **constraints**.
- Provides a **declarative model**: specify what must be satisfied, not how to search.
- CSP algorithms exploit structure for efficiency compared to uninformed search.
- General-purpose solvers apply across diverse tasks:
  - Scheduling (classes, exams, tasks)
  - Map coloring (regions with different colors)
  - N-Queens
  - Sudoku
- Foundation for reasoning about feasibility, optimization, and knowledge representation.

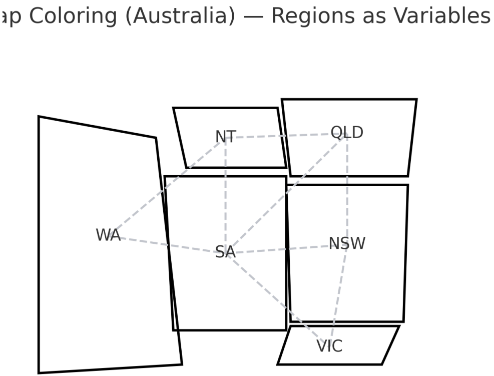
# Example: Scheduling

- Variables: tasks (or course sections); Domains: feasible time slots / rooms.
- Constraints: no overlap per resource (room/instructor), prerequisites, availability.
- Objective (optional): minimize gaps, balance load, maximize preferences.



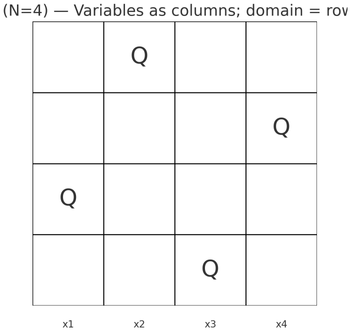
# Example: Map Coloring (Australia)

- Variables: regions $\{WA, NT, SA, QLD, NSW, VIC, TAS\}$.
- Domain: $\{red, green, blue\}$ (3-coloring variant).
- Constraints: adjacent regions must have different colors.



# Example: N-Queens (N=4)

- Variables: one per column $x_1, \ldots, x_4$; Domain: row index $\{1, \ldots, 4\}$.
- Constraints: no two queens share a row, column, or diagonal.
- Note: min-conflicts often solves large $N$ quickly (preview for local search).



# Example: Sudoku

- Variables: 81 cells $(r, c)$; Domain: $\{1, \ldots, 9\}$ (restricted by givens).
- Constraints: all-different in each row, column, and $3 \times 3$ subgrid.
- Variants: optimization (fewest conflicts), exact cover encodings, SAT reductions.

doku — 9×9 Grid with 3×3 Subgrid Constrair

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

# CSP: Formal Definition

**Constraint Satisfaction Problem (CSP)** is a triple $(X, D, C)$:

- **Variables** $X = \{X_1, X_2, \dots, X_n\}$.
- **Domains** $D = \{D_1, D_2, \dots, D_n\}$ where each $D_i$ is the set of allowable values for $X_i$.
- **Constraints** $C = \{C_1, C_2, \dots, C_m\}$, each $C_j$ specifies allowed combinations of values for a subset (its *scope*) of variables.

**Assignments and consistency**

- A (partial) assignment $\theta$ maps some variables to values in their domains.
- $\theta$ is *consistent* iff it does not violate any constraint whose scope is fully assigned.
- A *solution* is a *complete* assignment that satisfies all constraints.

# Map Coloring (Australia) as a CSP

**Variables**

$$X = \{\text{WA}, \text{NT}, \text{SA}, \text{Q}, \text{NSW}, \text{V}, \text{T}\}.$$

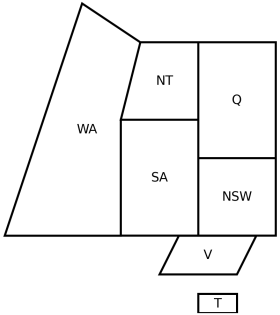
**Domains** (3 colors)

$$D_i = \{\text{Red}, \text{Green}, \text{Blue}\} \quad \text{for all } X_i \in X.$$

**Adjacency constraints** (neighboring regions must differ):

$$\text{WA} \neq \text{NT},\ \text{WA} \neq \text{SA},\ \text{NT} \neq \text{SA},\ \text{NT} \neq \text{Q},$$
$$\text{SA} \neq \text{Q},\ \text{SA} \neq \text{NSW},\ \text{SA} \neq \text{V},\ \text{Q} \neq \text{NSW},$$
$$\text{NSW} \neq \text{V}$$

(Tasmania **T** is isolated; no adjacency constraints.)



# States, Partial Assignments, and the Goal Test

**State representation**

- A *state* is a (possibly partial) assignment $\theta$ over $X$.
- Example: $\theta = \{\text{WA} = \text{Red},\ \text{NT} = \text{Green}\}$.

**Consistency (a.k.a. feasibility)**

- A partial state is *consistent* if no constraint is violated by currently assigned variables.
- Consistency depends only on the scopes that are fully assigned in $\theta$.

**Goal test**

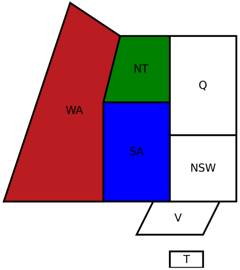
- *Goal*: a *complete* assignment ($|\text{dom}(\theta)| = |X|$) that satisfies *all* constraints in $C$.
- For Australia: all seven regions assigned colors, and every adjacent pair differs.

# Partial Assignment Examples (Australia)

**Example A — consistent (partial)**

$\theta_A = \{\text{WA} = \text{Red},\ \text{NT} = \text{Green},\ \text{SA} = \text{Blue}\}$

Checks: WA≠NT, WA≠SA, NT≠SA $\Rightarrow$ *no violations so far*.

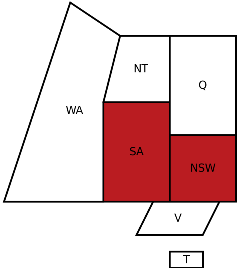


# Partial Assignment Examples (Australia)

**Example B — inconsistent (partial)**

$\theta_B = \{\text{SA} = \text{Red},\ \text{NSW} = \text{Red}\}$

Check: SA$\neq$NSW is violated $\Rightarrow$ *inconsistent*.

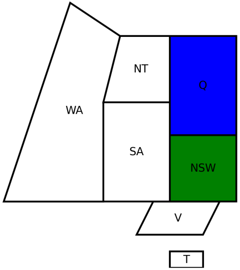


# Partial Assignment Examples (Australia)

**Example C — consistent but incomplete**

$\theta_C = \{\text{Q} = \text{Blue},\ \text{NSW} = \text{Green}\}$

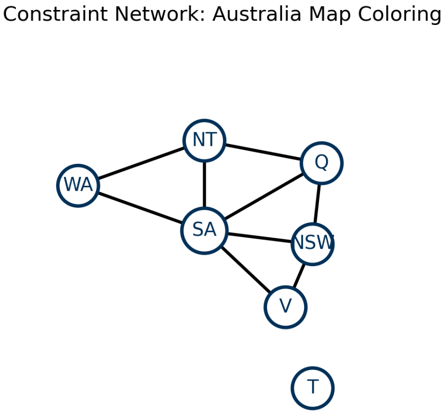
Checks: Q$\neq$NSW $\Rightarrow$ satisfied; variables remaining: WA, NT, SA, V, T.



# Visualizing CSPs: Constraint Networks

- A CSP can be represented as a **constraint network**:
  - **Nodes**: Variables
  - **Edges**: Constraints between variables
- Makes the structure of the problem explicit.
- Useful for reasoning about:
  - Constraint tightness
  - Variable connectivity
  - Ordering heuristics

# Visualizing CSPs: Constraint Networks



# Constraint Orderings: Map Coloring

- Variable ordering affects search efficiency.
- Example: Australia map coloring
  - Variables: $\{$WA, NT, SA, Q, NSW, V, T$\}$
  - Domain: $\{$Red, Green, Blue$\}$
  - Constraints: Adjacent regions $\neq$ color
- Two possible orderings:
  1. Start with Tasmania (low connectivity) $\rightarrow$ poor choice
  2. Start with South Australia (high connectivity) $\rightarrow$ better choice

# Constraint Orderings: Map Coloring

Constraint Orderings: Australia Map Coloring

Ordering A (poor): low connectivity first

Ordering B (better): start at high-degree SA

# Backtracking Search for CSPs

- Basic search method for CSPs.
- Builds assignments incrementally.
- At each step:
  1. Choose an unassigned variable.
  2. Assign it a value consistent with prior assignments.
  3. If conflict: backtrack.
- Depth-first, systematic, but may be exponential.

# CSP Backtracking (with MRV/Degree & LCV hooks)

**Algorithm 1** Backtracking Search for CSPs

**Require:** variables $X = \{X_1, \ldots, X_n\}$, domains $D(X_i)$, constraints $C$

**Ensure:** a complete assignment $\mathcal{A}$ satisfying all $C$ or FAILURE

```
function BACKTRACKING-SEARCH(X, D, C)
    return BACKTRACK({}, X, D, C)                      ▷ start with empty assignment
end function
function BACKTRACK(𝒜, X, D, C)
    if 𝒜 assigns all variables in X then return 𝒜
    end if
    X_i ← SELECT-UNASSIGNED-VARIABLE(𝒜, X, C)          ▷ MRV then Degree
    for all v ∈ ORDER-DOMAIN-VALUES(X_i, 𝒜, D, C) do    ▷ LCV
        if CONSISTENT(X_i ← v, 𝒜, C) then
            𝒜′ ← 𝒜 ∪ {X_i ↦ v}
            result ← BACKTRACK(𝒜′, X, D, C)
            if result ≠ FAILURE then return result
            end if
        end if
    end for
    return FAILURE
end function
```

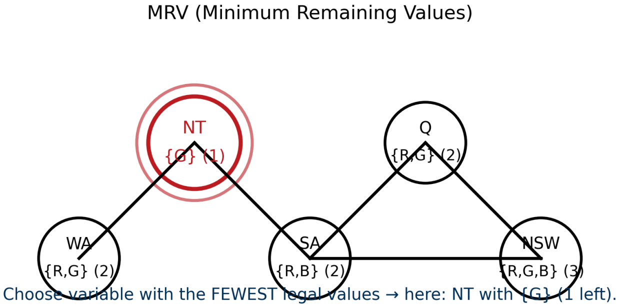
# CSP Backtracking (with MRV/Degree & LCV hooks)

**Algorithm 2** Backtracking Search for CSPs

**function** SELECT-UNASSIGNED-VARIABLE($\mathcal{A}$, $X$, $C$)
    $U \leftarrow \{X_i \in X \mid X_i$ unassigned in $\mathcal{A}\}$
    **return** $X_i \in U$ with *minimum* $|\text{legal\_values}(X_i \mid \mathcal{A}, C)|$ **(MRV),** tie-break by *maximum* degree w.r.t. other vars in $U$
**end function**
**function** ORDER-DOMAIN-VALUES($X_i, \mathcal{A}, D, C$)
    **return** values of $D(X_i)$ sorted by *least* number of values ruled out in neighbors **(LCV)**
**end function**
**function** CONSISTENT($X_i \leftarrow v$, $\mathcal{A}$, $C$)
    **return** TRUE iff $\forall$ constraint $c \in C$ over vars in $\mathcal{A} \cup \{X_i\}$, the partial assignment satisfies $c$
**end function**

# Heuristic: Minimum Remaining Values (MRV)

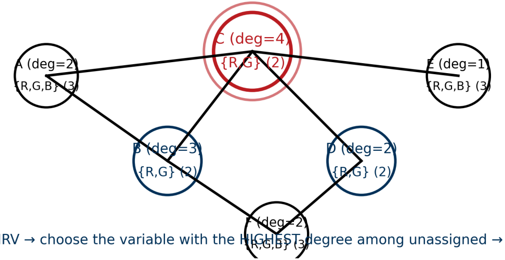
- Also called the **most constrained variable.**
- Choose the variable with the fewest legal values left.
- Intuition: Fail fast — detect dead ends early.



# Heuristic: Degree Heuristic

- Tie-breaker for MRV.
- Choose variable involved in the largest number of constraints on other unassigned variables.
- Intuition: Assign the most “constraining” variable first.

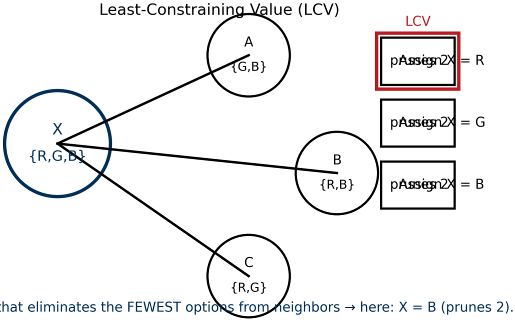
Degree Heuristic (tie-breaker for MRV)




Tie on MRV → choose the variable with the HIGHEST degree among unassigned → here: C.

# Heuristic: Least-Constraining Value

- When selecting a value for a variable, prefer the one that leaves the most options open for others.
- Intuition: Reduce branching factor by preserving flexibility.

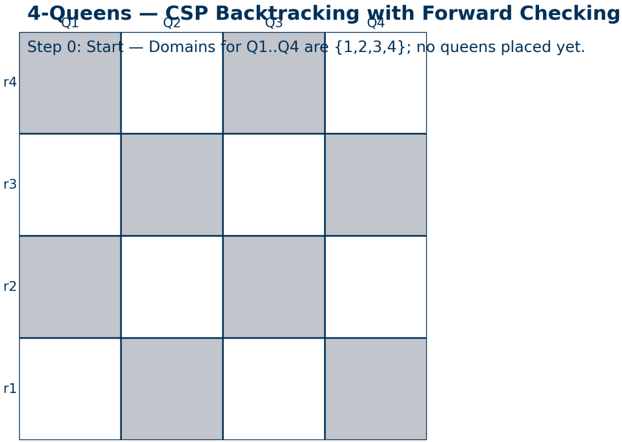


# Local Search for CSPs: Motivation

# Min-Conflicts Heuristic (N-Queens Example)

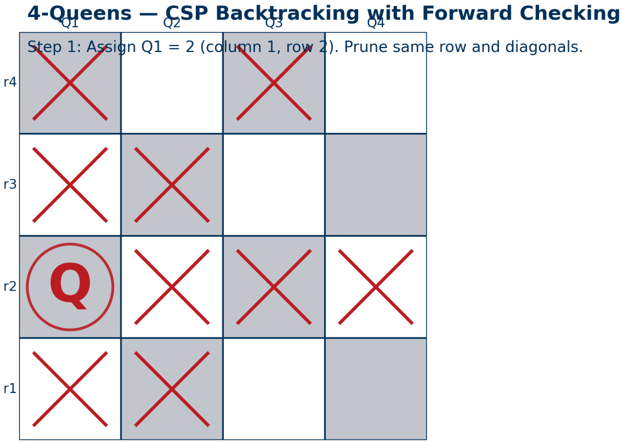
# Local Search: Properties & When to Use

# Example Walkthrough: 4-Queens (Step 0)



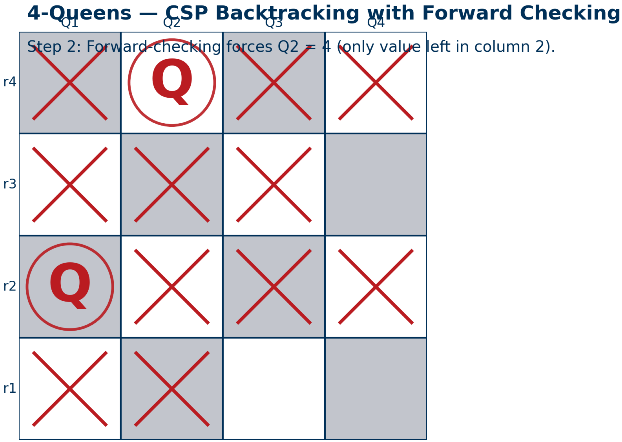

Start — domains for all variables are $\{1,2,3,4\}$.

# Example Walkthrough: 4-Queens (Step 1)



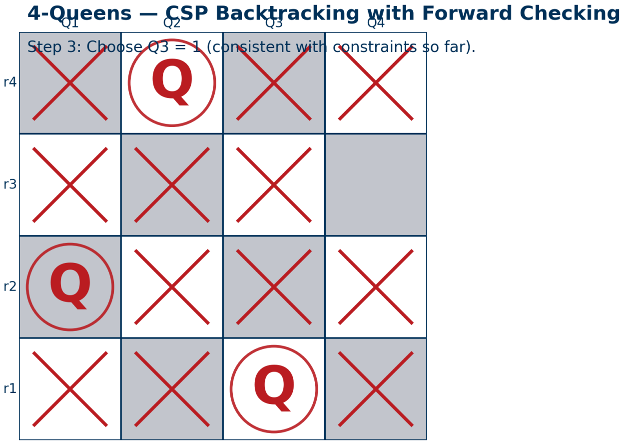

Assign $Q_1 = 2$; prune same row/diagonals via forward checking.

# Example Walkthrough: 4-Queens (Step 2)



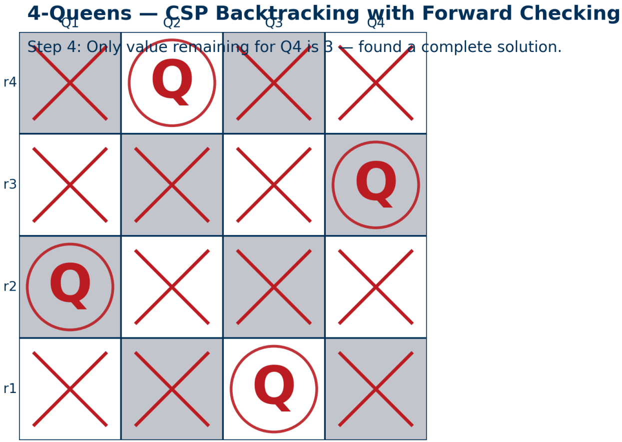

Forward-checking forces $Q_2 = 4$ (only value left).

# Example Walkthrough: 4-Queens (Step 3)




Choose $Q_3 = 1$ consistent with constraints so far.

# Example Walkthrough: 4-Queens (Step 4)




Only value remaining for $Q_4$ is $3$ — solution $(2, 4, 1, 3)$.

# Summary: CSP Strategies & Advantages

- **Model once, solve many:** Variables, domains, constraints unify diverse problems (map coloring, n-queens, scheduling).
- **Search with pruning:** Backtracking + **forward checking** and **constraint propagation** (e.g., arc consistency) cut the search dramatically.
- **Heuristics matter: MRV** (min-remaining-values), **degree**, and **least-constraining-value** guide choices effectively.
- **Local search options:** Min-conflicts excels on large/loose CSPs; often finds solutions quickly from random starts.
- **Tradeoffs:** Completeness vs. speed; stronger propagation costs more per step but reduces backtracking.
- **Takeaway:** Well-chosen representations + propagation + heuristics $\Rightarrow$ tractable solutions for large CSPs.