Constraint Satisfaction Problems & solutions

CS 220 Data, analytics, TA session

Yufan Liu

Computer Science Program, KAUST

Nov 30, 2024

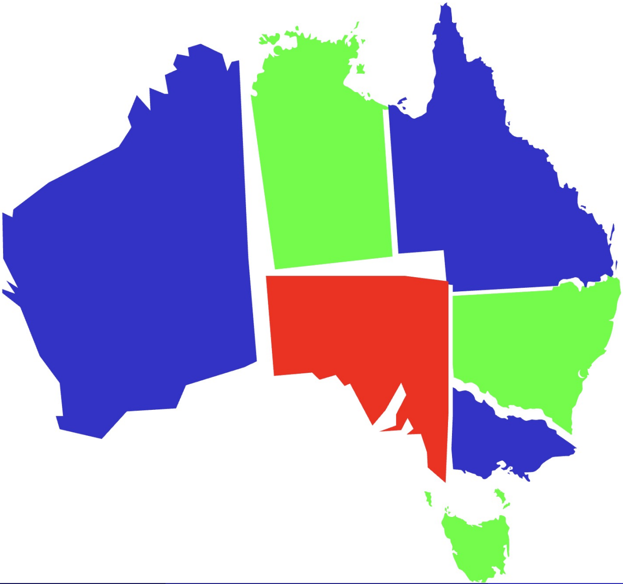
Recap: CSP definition

- **Components:**
  - A set of variables: $x_1, x_2, \ldots, x_n$
  - A set of constraints: $C_1, C_2, \ldots, C_m$
  - Each variable x_i has a non-empty domain D_i of possible values.
  - Each constraint C_i:
    - Involves a subset of variables.
    - Specifies the allowable combinations of values for that subset.
- **State:**
  - Defined by an assignment of values to some or all variables.
  - 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.
- **Solution:**
  - A complete assignment that satisfies all constraints.

Recap: CSP in searching

- **Initial State:**
  - The empty assignment, in which all variables are unassigned.
- **Successor Function:**
  - A value can be assigned to any unassigned variable.
  - The assignment must not conflict with previously assigned variables.
- **Goal Test:**
  - The current assignment is complete (all variables are assigned).
- **Path Cost:**
  - A constant cost (e.g., 1) is incurred for each step.

Map Coloring Problem as a CSP



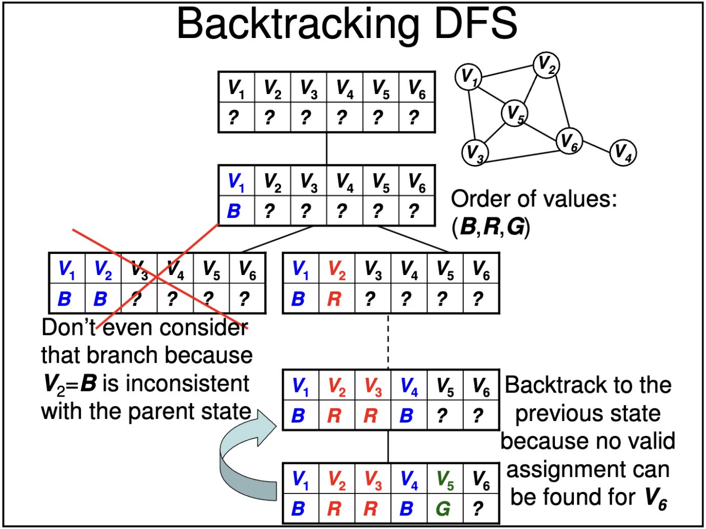
Map Coloring Problem as a CSP

- **Problem Description:**
  - Given a map consisting of multiple regions, some of which are adjacent.
  - Each region must be assigned a color.
  - Adjacent regions cannot share the same color.
- **CSP Representation:**
  - **Variables:** Each region is a variable x_i, representing the color of the region.
  - **Domains:** Each variable x_i has a domain D_i of possible colors (e.g., $\{$red, blue, green$\}$).
  - **Constraints:** For any two adjacent regions x_i and x_j, $x_i \neq x_j$.
- **Solution:**
  - A valid assignment of colors to all regions such that no adjacent regions share the same color.

Solutions for CSP

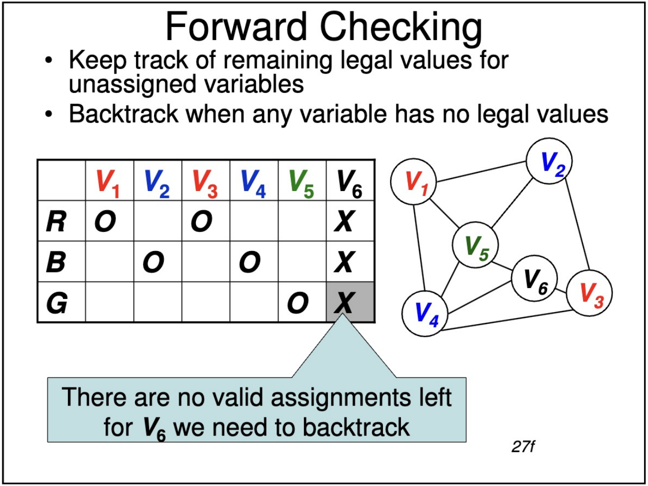
- **Plain Search**: DFS or BFS
  - Use a brute-force approach to evaluate each potential solution.
  - $n!d^n$ possible nodes, but n^d assignments.
- **Backtracking Search**: Systematically explore possible assignments and backtrack when constraints are violated.
  - Basic strategy for solving CSPs.
  - Prunes the search space by abandoning paths that lead to invalid solutions.
- **Heuristic Search**: Improve efficiency by prioritizing the search order.
  - Use heuristics such as Minimum Remaining Values (MRV) or Least Constraining Value (LCV).

Backtracking



Backtracking

- Uninformed search.
- Can be improved by forward checking or consistency.



Arc consistency (AC-3)

Goal: Ensure each value of one variable has a consistent value in related variables.

Steps:

1. **Initialize:** Add all arcs (X, Y) to a queue Q.
2. **Process Arcs:**
   - While Q is not empty:
     1. Remove an arc (X, Y) from Q.
     2. Check if any value in $D(X)$ violates the constraint with $D(Y)$:

        $$\text{Remove } x \in D(X) \text{ if no } y \in D(Y) \text{ satisfies } C(X, Y).$$

     3. If $D(X)$ changes, re-add all related arcs (Z, X) to Q.
3. **Terminate:** When Q is empty, domains are arc-consistent. If a domain D(X) is empty, this searching has no solutions.

Example with AC-3 Algorithm

Problem:

- **Variables:** X, Y, Z
- **Domains:** $D(X) = \{1, 2, 3\}, D(Y) = \{2, 3\}, D(Z) = \{1, 2, 3\}$
- **Constraints:** $X < Y, Y \neq Z$

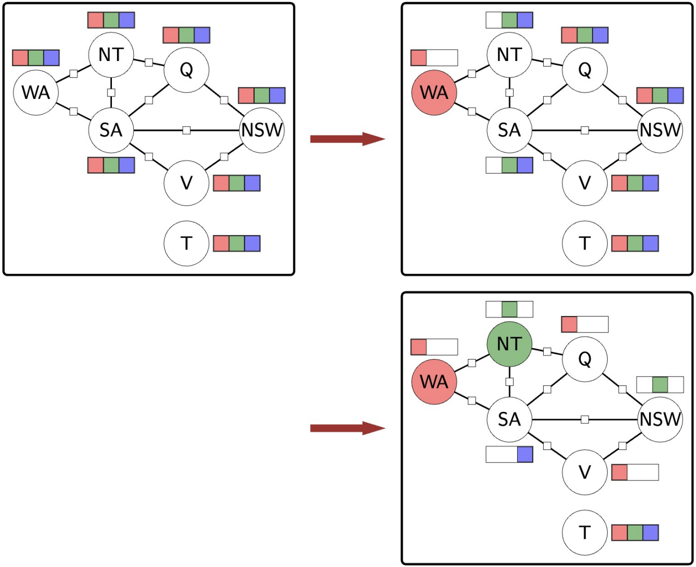
AC-3 Steps:

1. Initialize $Q = \{(X, Y), (Y, X), (Y, Z), (Z, Y), (X, Z), (Z, X)\}$
2. Process (X, Y): Remove 3 from $D(X)$ ($X < Y$ fails for 3).
3. Update $D(X) = \{1, 2\}$, recheck related arcs.
4. Continue until $Q = \emptyset$.

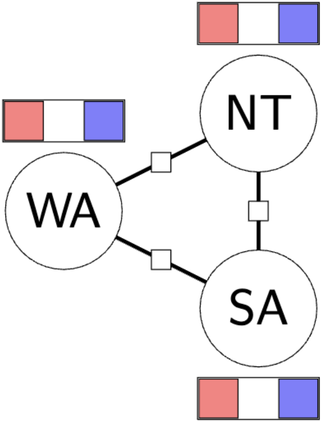
Result:

$$D(X) = \{1, 2\}, \quad D(Y) = \{2, 3\}, \quad D(Z) = \{1, 2, 3\}$$

AC-3 example: earlier than forward checking



AC-3 example



K-consistency: general formulation

- **Node consistency**: single value constraints.
- **Arch consistency**: 2-consistency.
- **Path consistency**: compatibility of three-variables.
- **K-consistency**: high-order consistency.

Heuristic search

Variable Selection Heuristics:

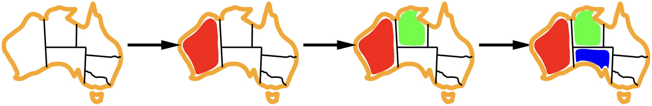
- **Minimum Remaining Value (MRV):**
  - *Idea:* Choose the variable with the fewest legal values remaining.
  - *Reason:* Reduces conflict potential and reveals unsolvable branches earlier.
- **Degree Heuristic:**
  - *Idea:* When MRV ties, choose the variable with the most constraints on other unassigned variables.
  - *Reason:* Solves critical variables earlier, reducing complexity for others.

Value Selection Heuristics:

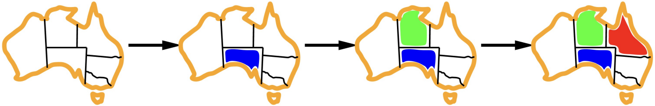
- **Least Constraining Value (LCV):**
  - *Idea:* Select the value that imposes the fewest restrictions on the legal values of other variables.
  - *Reason:* Minimizes future conflicts, increasing the likelihood of successful assignment.

Variable selection heuristics

- Minimum Remaining Values (MRV)

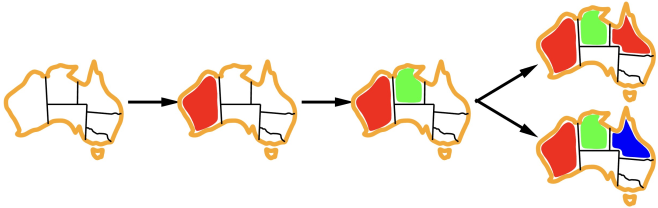


- Degree Heuristic



Value selection heuristics

- Least Constraining Value (LCV)



References

- CS 220 slides, Xin Gao, KAUST
- CS 188 slides, Stuart Russell, UC Berkeley
- 15-381 slides, Martial Hebert and Mike Lewicki, CMU
- CS 221 slides and an online lecture, Percy Liang, Stanford