COMP7015 Artificial Intelligence (S1, 2024-25)

Lecture 3 Part A: Constraint Satisfaction Problems

Instructor: Dr. Kejing Yin (cskjyin@hkbu.edu.hk)

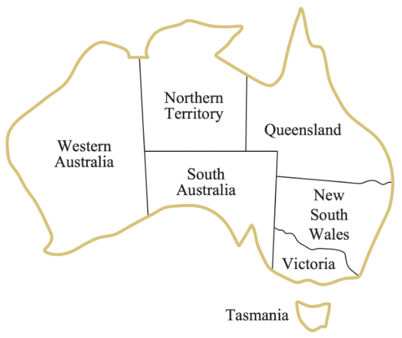
Department of Computer Science
Hong Kong Baptist University

September 20, 2024

Constraint Satisfaction Problems

- Recap
- Variable and Value Ordering
- Live Demo

Recap: Formulating Constraint Satisfaction Problems (CSP)



Three components of a CSP:

- A set of **variables**: $\mathcal{X} = \{\mathsf{WA}, \mathsf{NT}, \mathsf{SA}, \mathsf{Q}, \mathsf{NSW}, \mathsf{V}, \mathsf{T}\}$.
- Each variable has a **domain** of possible values: $\mathcal{D}_i = \{\text{red}, \text{green}, \text{blue}\} \quad \forall i \in \mathcal{X}$
- A set of **constraints**:

$$\mathcal{C} = \{\mathsf{WA} \neq \mathsf{NT}, \mathsf{WA} \neq \mathsf{SA}, \mathsf{NT} \neq \mathsf{SA}, \mathsf{NT} \neq \mathsf{Q}, \mathsf{SA} \neq \mathsf{Q}, \mathsf{SA} \neq \mathsf{NSW}, \mathsf{SA} \neq \mathsf{V}, \mathsf{Q} \neq \mathsf{NSW}, \mathsf{NSW} \neq \mathsf{V}\}$$

A **solution** is an assignment to all variables such that no constraint is violated. For example:

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

Recap: Formulating Sudoku as a CSP

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

- Variables: $\mathcal{X} = \{A1, \ldots, A9, \ldots, I1, \ldots, I9\}$
- Domains: $\mathcal{D}_i = \{1, 2, 3, 4, 5, 6, 7, 8, 9\} \;\; \forall i \in \mathcal{X}$
- Constraints:
  - $A3 = 3,\; A5 = 2,$ ...
    (existing numbers must match)
  - $\text{AllDiff}(A1, A2, \ldots, A9)$, ...
    (each row has all different values)
  - $\text{AllDiff}(A1, B1, \ldots, I1)$, ...
    (each column has all different values)
  - $\text{AllDiff}(A1, A2, A3, B1, B2, B3, C1, C2, C3)$, ...
    (each 3x3 region has all different values)

Recap: Another Formulation of Sudoku as a CSP

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

- Variables:
  $\mathcal{X} = \{A1, A2, \cancel{A3} \ldots, A9, \ldots, I1, \ldots, I9\}$
  (only consider empty cells as variables)
- Domains: $\mathcal{D}_1 = \{1, \cancel{2}, \cancel{3}, 4, 5, \cancel{6}, \cancel{7}, \cancel{8}, \cancel{9}\}$, ...
  (only consider values consistent with existing cells)
- Constraints:
  - $A3 = 3,\ A5 = 2,$ ...
    (existing numbers must match)
  - $\mathrm{AllDiff}(A1, A2, \ldots, A9)$, ...
    (each row has all different values)
  - $\mathrm{AllDiff}(A1, B1, \ldots, I1)$, ...
    (each column has all different values)
  - $\mathrm{AllDiff}(A1, A2, A3, B1, B2, B3, C1, C2, C3)$, ...
    (each 3x3 region has all different values)

Recap: Backtracking with Forward Checking

Forward Checking:

- After selecting each assignment, remove any values of neighboring domains that are inconsistent with the new assignment.
- Terminate search when any variable has no legal values.

Example: Variables $A, B, C \in \{1, 2, 3\}$. Constraints: $A > B, B \neq C, A \neq C$.

step 1:

{}

A=1

✗

$\mathcal{D}_B = \{\cancel{1}, \cancel{2}, \cancel{3}\}$

step 2:

{}

A=1 A=2

✗

$\mathcal{D}_B = \{1, \cancel{2}, \cancel{3}\}$,
$\mathcal{D}_C = \{1, \cancel{2}, 3\}$

step 3:

{}

A=1 A=2

✗ B=1

$\mathcal{D}_B = \{1, \cancel{2}, \cancel{3}\}$,
$\mathcal{D}_C = \{\cancel{1}, \cancel{2}, 3\}$

step 4:

{}

A=1 A=2

✗ B=1

C=3

✓

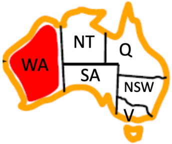
$\mathcal{D}_B = \{1, \cancel{2}, \cancel{3}\}$,
$\mathcal{D}_C = \{\cancel{1}, \cancel{2}, 3\}$

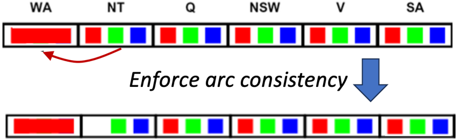
Recap: Backtracking with Arc Consistency

📖 Arc Consistency

Variable X_i is *arc-consistent* with respect to another variable X_j if for every value in the current domain $\mathcal{D}_i$, there is some value in the domain $\mathcal{D}_j$ that satisfies the binary constraint on the arc (X_i, X_j).

Example:





EnforceArcConsistency(NT, WA):

- If NT $=$ blue or NT $=$ green: binary constraint satisfied (WA$\neq$NT), consistent.
- If NT $=$ red: constraint violated, delete red from the domain of NT to make NT arc consistent.

Recap: Backtracking Search Algorithm with Constraint Propagation

The Backtracking Search Algorithm (with Constraint Propagation)

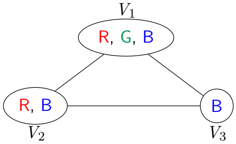
```
function Backtrack(assignment 𝒜, domains 𝒟)
    if assignment 𝒜 is complete then return 𝒜;
    select an unassigned variable X_i;
    order values for the variable X_i;
    for value v in that order do
        if v is not consistent with 𝒜 then continue;
        assign X_i = v;
        𝒟' ← propagate constraints ; // forward checking or arc consistency
        if any variable has an empty domain in 𝒟' then continue;
        Backtrack(𝒜 ⋃ {X_i : v}, 𝒟')
    end
end
```

Constraint propagation: Using constraints to reduce the number of valid values for variables.

Remaining questions: 1) How to select variable? 2) How to order values?

Variable Ordering: Minimum-Remaining-Value (MRV) Heuristic

For the following problem, which variable will you assign first?



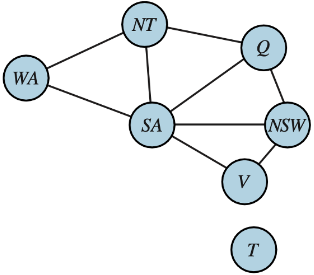
- Variables: $\mathcal{X} = \{V_1, V_2, V_3\}$.
- Domains: $\mathcal{D}_1=\{R,G,B\}$, $\mathcal{D}_2=\{R,B\}$, $\mathcal{D}_3=\{B\}$.
- Constraints: adjacent variables must have different colors.

📖 Minimum-Remaining-Value (MRV) Heuristic

- Choosing the variable with the fewest "legal" values.
- In this example, we assign V_3 first, then V_2, and finally V_1.
- Also called the "most constrained variable" or "fail-first" heuristic.
- Fewer "legal" values $\Rightarrow$ more likely to cause a failure soon $\Rightarrow$ prune the search tree.

Variable Ordering: Degree Heuristic

For the Australian map coloring problem, which variable will you assign first?



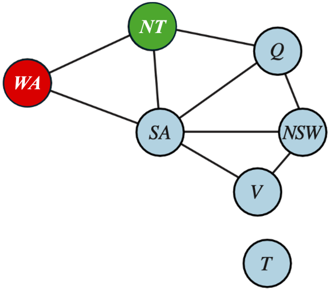
📖 Degree Heuristic

- Choosing the variable with the highest degree.
- "Degree": number of edges connecting to the node in a graph, e.g., $deg(\text{SA}) = 5$, $deg(\text{WA}) = 2$, $deg(\text{T}) = 0$.
- In this example, we assign SA first.

- The minimum-remaining-value heuristic is usually a more powerful guide.
- The degree heuristic can be useful as a tie-breaker.

Value Ordering: Least-Constraining-Value (LCV) Heuristic

Suppose we color WA as red, NT as green, and we choose Q to assign next.
What value should we assign? red or blue?



📖 Least-Constraining-Value (LCV) Heuristic

- Choosing the value that rules out the *fewest* choices for the neighboring variables in the constraint graph.
- In this example, we assine Q=red first. *(blue eliminates the last legal value left for SA.)*
- It attempts to leave the maximum flexibility for subsequent variable assignments.

Variable and Value Ordering: Summary

- For variable selection: “fail-first” minimum-remaining-value (MRV) heuristic.
- For value ordering: “fail-last” least-constraining-value (LCV) heuristic.

Why should variable selection be fail-first, but value selection be fail-last?

- *All* variable has to be assigned eventually! So, if an assignment is eventually going to fail, the sooner it fails, the more we prune the search tree.
- However, each variable only need to take *one* value. So, we choose the value that is most likely to lead to a solution.

Live Demos for Australian Map Coloring

- [Backtrack Search without Constraint Propagation]
- [Backtrack Search with Forward Checking]

Live Demos for Sudoku

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

- [Backtrack Search without Constraint Propagation]
- [Backtrack Search with Forward Checking]
- [Variable Selection Heuristic (MRV)]
- [Value Ordering Heuristic (LCV)]

After-class programming exercise:

- Implement AC3 and compare AC3 with forward checking.