**Constraint Satisfaction**

CSE 473  
University of Washington

---

**Today:**  
**Constraint Satisfaction Problems**

- Definition  
  Factoring state spaces
- Variable-ordering heuristics
- Backtracking policies
- Preprocessing algorithms

---

**Constraint Satisfaction**

- Kind of *search* in which  
  States are *factored* into sets of variables  
  Search = assigning values to these variables  
  Structure of space is encoded with constraints
- Backtracking-style algorithms work  
  E.g. DFS for SAT (i.e. DPLL)
- But other techniques add speed  
  Propagation  
  Variable ordering  
  Preprocessing

---

**Chinese Food as Search**

- States?
- Operators?
- Start state?  
  Suppose every meal for n people has n dishes plus soup
- Goal states?
Chinese Food as Search

- **States?**
  - Partially specified meals

- **Operators?**
  - Add, remove, change dishes

- **Start state?**
  - Null meal

- **Goal states?**
  - Meal meeting certain conditions

  Two non-peanut dishes, with at least one of the others non-spicy.

Factoring States

- Break the state into its independent components:
  
  \[ \text{State} = (\text{dish}_1, \ldots, \text{dish}_n, \text{soup}) \]

  **Becomes variables:**

  - Soup =
  - Dish 1 =
  - Dish 2 =
  - ...
  - Dish n =

Factoring States

- Encode the structure of the problem as constraints

- Often this structure is the description of the goal state:

  Goal condition = “Two peanut-free dishes; of the others, at least one must be non-spicy”

How to express this as constraints?

Chinese Constraint Network
### Binary Constraint Network

- **Set of** \( n \) **variables**: \( x_1 \ldots x_n \)
- **Value domains** for each variable: \( D_1 \ldots D_n \)
- **Set of binary constraints** (also “relations”)
  - Each binary constraint specifies which pairs \((x_i, x_j)\) are consistent
  - \( R_{ij} \subseteq D_i \times D_j \)

**Partial assignment** of values = tuple of pairs
- \( \{(x, a)\ldots\} \) means variable \( x \) gets value \( a \)...

**Tuple** = **consistent** if all constraints satisfied
- **Tuple** = **full solution** if consistent + has all vars

\[
\text{Tuple } \{(x_i, a_i) \ldots (x_j, a_j)\} = \text{consistent w/ a set of vars } \{x_m \ldots x_n\}
\]

\[
\text{iff } \exists a_m \ldots a_n \text{ such that} \\
\{(x_i, a_i)\ldots(x_j, a_j), (x_m, a_m)\ldots(x_n, a_n)\} = \text{consistent}
\]

### CSPs in the Real World

- Scheduling space shuttle repair
- Airport gate assignments
- Transportation Planning
- Supply-chain management
- Computer configuration
- Diagnosis
- UI optimization
- Etc...

### CSP Example: Cryptarithmetic

- **State Space**
  - Set of states
  - Operators [and costs]
  - Start state
  - Goal states

- **Variables**?
- **Domains** (variable values)?
- **Constraints**?

\[
\begin{align*}
\text{TWO} + \text{TWO} &= \text{FOUR} \\
102 + 102 &= 204
\end{align*}
\]
CSP Example: Cryptarithmetic

- State Space
  - Set of states
  - Operators [and costs]
  - Start state
  - Goal states

  \[
  \begin{array}{c}
  \text{T W O} \\
  + \text{T W O} \\
  \text{F O U R}
  \end{array}
  \]

CSP Example: Classroom Scheduling

- Variables?
- Domains (variable values)?
- Constraints?

\[
\text{N Queens}
\]

- As a CSP?

N Queens

- Variables = board columns
- Domain values = rows
- \( R_{ij} = \{ (ai, a_j) : (ai \neq a_j) \land (|i-j| \neq |ai-a_j|) \} \)
  
  e.g. \( R_{12} = \{ (1,3), (1,4), (2,4), (3,1), (4,1), (4,2) \} \)

- \{ (x_1, 2), (x_2, 4), (x_3, 1) \} consistent with (x4)
- Shorthand: "{2, 4, 1} consistent with x4"
**CSP as a search problem?**

- What are states?
- What are the operators?
- Initial state?
- Goal test?

![Diagram of CSP states and operators](image.png)

**CSP as a search problem?**

- What are states?
  - (partial assignments)
- What are the operators?
  - (add/change a single-variable assignment)
- Initial state?
  - (empty assignment)
- Goal test?
  - (full & consistent assignment)

![Diagram of CSP states and operators](image.png)

**Chronological Backtracking (BT)**

(e.g., depth first search)

- Which queen to place next, and where?

![Diagram of queen placement](image.png)

**Variable and Value Ordering**

- Which queen to place next, and where?

![Diagram of queen placement](image.png)
Variable and Value Ordering

• Which queen to place next, and where?
• Most constrained variable / minimum remaining values (MRV)
• Least constraining value
• Highest degree (as tiebreaker)

Backjumping (BJ)

• Similar to BT, but
  more efficient when no consistent instantiation can be found for the current var
• Instead of backtracking to most recent var...
  BJ reverts to deepest var which was c-checked against the current var

Forward Checking (FC)

• Perform Consistency Check Forward
• Whenever a var is assigned a value
  Prune inconsistent values from
  As-yet unvisited variables
  Backtrack if domain of any var ever collapses

FC can’t detect that
(2, 5, 3) inconsistent with \{x_5, x_6\}
\rightarrow Arc consistency can fix this

CSP Summary

• CSPs are a special kind of problem
  states defined by values of a fixed set of variables
  goal test defined by constraints on variable values
• Forward checking prevents assignments that guarantee later failure
• Variable ordering and value selection heuristics help significantly
• Complexity depends on constraint graph: linear time if tree structured