What is Search For?

- **Planning:** sequences of actions
  - The *path to the goal* is the important thing
  - Paths have various costs, depths
  - Assume little about problem structure

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

Constraint Satisfaction Problems

- Standard search problems:
  - State is a "black box": arbitrary data structure
  - Goal test can be any function over states
  - Successor function can also be anything

- Constraint satisfaction problems (CSPs):
  - A special subset of search problems
  - State is defined by variables $X_i$ with values from a domain $D$ (sometimes $D$ depends on $i$)
  - Goal test is a set of constraints specifying allowable combinations of values for subsets of variables

- Making use of CSP formulation allows for optimized algorithms
  - Typical example of trading generality for utility (in this case, speed)

Constraint Satisfaction Problems are *structured* (factored) identification problems

- "Factoring" the state space
- Representing the state space in a knowledge representation

- Constraint satisfaction problems (CSPs):
  - A special subset of search problems
  - State is defined by variables $X_i$ with values from a domain $D$ (sometimes $D$ depends on $i$)
  - Goal test is a set of constraints specifying allowable combinations of values for subsets of variables

Previously

- Formulating problems as search
- Blind search algorithms
  - Depth first
  - Breadth-first (uniform cost)
  - Iterative deepening
- Heuristic Search
  - Best first
  - Beam (Hill climbing)
- A*, IDA*
- Heuristic generation
  - Exact soln to a relaxed problem
  - Pattern databases
- Local Search
  - Hill climbing, random moves, random restarts, simulated annealing
CSP Example: N-Queens

Formulation 1:
- Variables: $X_{ij}$
- Domains: $\{0, 1\}$
- Constraints:
  \[ \forall i, j, k \ (X_{ij}, X_{ik}) \in \{(0, 0), (0, 1), (1, 0)\} \]
  \[ \forall i, j, k \ (X_{ij}, X_{kj}) \in \{(0, 0), (0, 1), (1, 0)\} \]
  \[ \forall i, j, k, l \ (X_{ij}, X_{il}, X_{kj}, X_{kl}) \in \{(0, 0), (0, 1), (1, 0)\} \]
  \[ \sum_{ij} X_{ij} = N \]

Formulation 2:
- Variables: $Q_k$
- Domains: $\{1, 2, 3, \ldots N\}$
- Constraints:
  Implicit: $\forall i, j$ non-threatening($Q_i, Q_j$)
  Explicit: $(Q_1, Q_2) \in \{(1, 3), (1, 4), \ldots\}$

CSP Example: Sudoku

Variables:
- Each (open) square
- Domains: $\{1, 2, \ldots, 9\}$
- Constraints:
  9-way alldiff for each row
  9-way alldiff for each column
  9-way alldiff for each region (or can have a bunch of pairwise inequality constraints)

Propositional Logic

- Variables: propositional variables
- Domains: $\{T, F\}$
- Constraints: logical formula

CSP Example: Map Coloring

- Variables: WA, NT, Q, NSW, V, SA, T
- Domains: $D = \{\text{red, green, blue}\}$
- Constraints: adjacent regions must have different colors
  Implicit: WA $\neq$ NT
  Explicit: (WA, NT) $\in \{\text{red, green, red, blue, blue, green, \ldots}\}$
- Solutions are assignments satisfying all constraints, e.g.:
  (WA$\rightarrow$red, NT$\rightarrow$green, Q$\rightarrow$red, NSW$\rightarrow$green, V$\rightarrow$red, SA$\rightarrow$blue, T$\rightarrow$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 subproblem!

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:
  \( \text{all}(F, T, U, W, R, O) \)
  \( O + O = R + 10 \cdot X_3 \)

Chinese Constraint Network

- Total Cost $\leq 540$

Real-World CSPs

- Assignment problems: e.g., who teaches what class
- Timetabling problems: e.g., which class is offered when and where?
- Hardware configuration
  - Gate assignment in airports
  - Space Shuttle Repair
- Transportation scheduling
- Factory scheduling
- ... lots more!

Example: The Waltz Algorithm

- The Waltz algorithm is for interpreting line drawings of solid polyhedra as 3D objects
- An early example of an AI computation posed as a CSP

Waltz on Simple Scenes

- Assume all objects:
  - Have no shadows or cracks
  - Three-faced vertices
  - "General position": no junctions change with small movements of the eye.
- Then each line on image is one of the following:
  - Boundary line (edge of an object) (+) with right hand of arrow denoting "solid" and left hand denoting "space"
  - Interior convex edge (-)
  - Interior concave edge (×)
Legal Junctions

- Only certain junctions are physically possible
- How can we formulate a CSP to label an image?
- Variables: edges
- Domains: $\geq$, $\leq$, +, -
- Constraints: legal junction types

Slight Problem: Local vs Global Consistency

Varieties of CSPs

- Discrete Variables
  - Finite domains
    - Size of the domain, e.g., complete assignments
    - E.g., Boolean CSPs, including Boolean satisfiability (NP-complete)
  - Infinite domains (integers, strings, etc.)
    - E.g., job scheduling, variables are start/end times for each job
    - Linear constraints solvable, nonlinear undecidable

- Continuous variables
  - E.g., start/final times for Hubble Telescope observations
  - Linear constraints solvable in polynomial time by linear program methods (see CSE 521 for a bit of LP theory)

Varieties of CSP Variables

- Discrete variables
  - Linear constraints involve a single variable (equivalent to reducing domains), e.g.,
- Binary constraints involve pairs of variables, e.g.,
- Higher-order constraints involve 3 or more variables: e.g., cryptarithmetic column constraints
- Preferences (soft constraints):
  - E.g., red is better than green
  - Often representable by a cost for each variable assignment
  - Gives constrained optimization problems
  - (We’ll ignore these until we get to Bayes’ nets)

Solving CSPs
CSP as Search

- States
- Operators
- Initial State
- Goal State

Standard Depth First Search

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: assign a value to an unassigned variable
  - Goal test: the current assignment is complete and satisfies all constraints
  - We'll start with the straightforward, naïve approach, then improve it

Backtracking Search

- Backtracking search is the basic uninformed algorithm for solving CSPs
- Idea 1: One variable at a time
  - Variable assignments are commutative, so fix ordering
    - i.e., WA = red then NT = green is the same as NT = green then WA = red
  - Only need to consider assignments to a single variable at each step
- Idea 2: Check constraints as you go
  - i.e. consider only values which do not conflict previous assignments
  - Might have to do some computation to check the constraints
  - "Incremental goal test"
- Depth-first search with these two improvements is called backtracking search
- Can solve n-queens for n \approx 25

Backtracking Example
### Backtracking Search

- What are the choice points?

```haskell
-- functions BACKTRACKING [var] returns solution/failure
-- functions RECURSIVE-BACKTRACKING [assignment] returns solution/failure
-- if assignment is complete then return assignment!
--    for each value v in ORDER-DOMAIN-VALUES [var, assignment] do
--      if value is consistent with assignment from CONSTRAINTS [e, assignment] then
--        add [var = value] to assignment
--        result ← RECURSIVE-BACKTRACKING [assignment]
--      if result # failure then return result
--    remove [var = value] from assignment
-- return failure
```

### Improving Backtracking

- General-purpose ideas give huge gains in speed
  - Ordering:
    - Which variable should be assigned next?
    - In what order should its values be tried?
  - Filtering: Can we detect inevitable failure early?
  - Structure: Can we exploit the problem structure?

### Next: Constraint Satisfaction Problems - Part 2

- Kind of depth first search
- Is it complete?