A*: Robotics Examples

- Urban Challenge
  - Successor function?
  - Heuristic?

- Door Opening
  - Successor function?
  - Heuristic?

Other A* Applications

- Pathing / routing problems
- Resource planning problems
- Robot motion planning
- Language analysis
- Machine translation
- Speech recognition
- …

Announcements

- Project 1 due Friday 4:59pm
  - See course website for this week’s office hours, held by P1 GSI

- Lecture videos online

- Written Assignment Policy

Today

- CSPs

- Efficient Solution of CSPs
  - Search
  - Constraint propagation

- Local Search

Constraint Satisfaction Problems

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

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

  - Simple example of a formal representation language
  - Allows useful general-purpose algorithms with more power than standard search algorithms
Example CSP: Map-Coloring

- **Variables:** \( W, A, N, T, Q, N_S, W, V, S, A, T \)
- **Domain:** \( D = \{ \text{red, green, blue} \} \)
- **Constraints:** adjacent regions must have different colors
  
  \[
  W \neq N \\
  (W, N) \in \{ (\text{red, green}), (\text{red, blue}), (\text{green, red}) \ldots \}
  \]

- **Solutions** are assignments satisfying all constraints, e.g.,

  \[
  \{ W = \text{red}, N = \text{green}, Q = \text{red}, \\
  N_S = \text{green}, V = \text{red}, S = \text{blue}, T = \text{green} \}
  \]

Example CSP: N-Queens

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

Example CSP: N-Queens

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

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 CSP: Cryptarithmetic

- **Variables (circles):**
  
  \( F, T, U, W, R, O, X_1, X_2, X_3 \)
  
  \( + \)
  
  \( T \ W \ O \)

- **Domains:** \( \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \)
- **Constraints (boxes):**
  
  \( \text{alldiff}(F, T, U, W, R, O) \)

  \( O \mid O = R = 10 \cdot X_1 \)

  \( \ldots \)

Example CSP: Sudoku

- **Variables:**
  
  Each (open) square

  **Domains:**
  
  \( \{1, 2, \ldots, 9\} \)

  **Constraints**
  
  9-way alldiff for each column

  9-way alldiff for each row

  9-way alldiff for each region
Example CSP: The Waltz Algorithm

- The Waltz algorithm is for interpreting line drawings of solid polyhedra
- An early example of a computation posed as a CSP
- Look at all intersections
- Adjacent intersections impose constraints on each other

Varieties of CSPs

- Discrete Variables
  - Finite domains
    - Size d means O(d^n) 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-end state of a robot
  - Linear constraints solvable in polynomial time by LP methods
    (see cs170 for a bit of this theory)

Varieties of Constraints

- Varieties of Constraints
  - Unary constraints involve a single variable (equiv. to shrinking domains):
    \[ SA \neq \text{green} \]
  - Binary constraints involve pairs of variables:
    \[ SA \neq WA \]
  - 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)

Real-World CSPs

- Assignment problems: e.g., who teaches what class
- Timetabling problems: e.g., which class is offered when and where?
- Hardware configuration
- Transportation scheduling
- Factory scheduling
- Floorplanning
- Fault diagnosis
- … lots more!
- Many real-world problems involve real-valued variables…

What is Search For?

- Models of the world: single agents, deterministic actions, fully observed state, discrete state space
- Planning: sequences of actions
  - The path to the goal is the important thing
  - Paths have various costs, depths
  - Heuristics to guide, fringe to keep backups
- Identification: assignments to variables
  - The goal itself is important, not the path
  - All paths at the same depth (for some formulations)
  - CSPs are specialized for identification problems

Standard Search Formulation

- Standard search formulation of CSPs (incremental)
- Let’s start with the straightforward, dumb approach, then fix it
- States are defined by the values assigned so far
  - 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
- Simplest CSP ever: two bits, constrained to be equal
Search Methods

- What does BFS do?
  - [demo]
- What does DFS do?
  - [demo]
- What's the obvious problem here?
- What's the slightly-less-obvious problem?

Backtracking Search

- Idea 1: Only consider a single variable at each point
  - Variable assignments are commutative, so fix ordering
  - i.e., (WA = red then NT = green) same as (INT = green then WA = red)
  - Only need to consider assignments to a single variable at each step
  - How many leaves are there?
- Idea 2: Only allow legal assignments at each point
  - i.e., consider only values which do not conflict previous assignments
  - Might have to do some computation to figure out whether a value is ok
  - "Incremental goal test"
- Depth-first search for CSPs with these two improvements is called backtracking search (useless name, really)
  - [DEMO]
- Backtracking search is the basic uninformed algorithm for CSPs
  - Can solve n-queens for n = 25

Backtracking Search

Function BACKTRACKING-SEARCH(assignment, goals) returns solution or failure
return RECURSIVE-BACKTRACKING(assignment, goals)

function RECURSIVE-BACKTRACKING(assignment, goals) returns solution or failure
if assignment is complete then return assignment
var ← SELECT-UNASSIGNED-VARIABLE(assignment, goals)
for each value in ORDER-DOMAINS(var, assignment, goals) do
  if value is consistent with assignment given CONSTRAINTS(var) then
    new ← ASSIGN(var, value) to assignment
    if result = failure then return result
    new ← RECURSIVE-BACKTRACKING(assignment, goals)
    if result is solution then return result
  end if
end for
return failure

- What are the choice points?

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?

Minimum Remaining Values

- Minimum remaining values (MRV):
  - Choose the variable with the fewest legal values
- Why min rather than max?
- Also called "most constrained variable"
- Also called "fail-fast" ordering

Degree Heuristic

- Tie-breaker among MRV variables
- Degree heuristic:
  - Choose the variable participating in the most constraints on remaining variables
- Why most rather than fewest constraints?
Least Constraining Value

- Given a choice of variable:
  - Choose the least constraining value
  - The one that rules out the fewest values in the remaining variables
  - Note that it may take some computation to determine this!
- Why least rather than most?
- Combining these heuristics makes 1000 queens feasible

Filtering: Forward Checking

- Idea: Keep track of remaining legal values for unassigned variables (using immediate constraints)
- Idea: Terminate when any variable has no legal values

Consistency of An Arc

- An arc $X \rightarrow Y$ is consistent iff for every $x$ in the tail there is some $y$ in the head which could be assigned without violating a constraint

Establishing Arc Consistency

- Simplest form of propagation makes each arc consistent
  - $X \rightarrow Y$ is consistent iff for every $x$ there is some allowed $y$
- If $X$ loses a value, neighbors of $X$ need to be rechecked!
- Arc consistency detects failure earlier than forward checking
- What’s the downside of arc consistency?
- Can be run as a preprocessor or after each assignment

Arc Consistency of a CSP

- If an arc loses a value, neighbors of the arc need to be rechecked!
- Arc consistency detects failure earlier than forward checking
- What’s the downside of arc consistency?
- Can be run as a preprocessor or after each assignment
Limitations of Arc Consistency

- After running arc consistency:
  - Can have one solution left
  - Can have multiple solutions left
  - Can have no solutions left (and not know it)

K-Consistency

- Increasing degrees of consistency
  - 1-Consistency (Node Consistency): Each single node's domain has a value which meets that node's unary constraints
  - 2-Consistency (Arc Consistency): For each pair of nodes, any consistent assignment to one can be extended to the other
  - K-Consistency: For each k nodes, any consistent assignment to k-1 can be extended to the kth node.

- Higher k more expensive to compute

Strong K-Consistency

- Strong k-consistency: also k-1, k-2, ... 1 consistent
- Claim: strong n-consistency means we can solve without backtracking!
- Why?
  - Choose any assignment to any variable
  - Choose a new variable
  - By 2-consistency, there is a choice consistent with the first
  - Choose a new variable
  - By 3-consistency, there is a choice consistent with the first 2
  - ...

- Lots of middle ground between arc consistency and n-consistency! (e.g. path consistency)