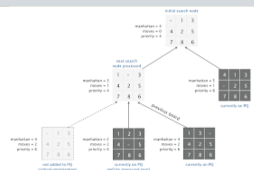
Artificial Intelligence

Class 3: Search (Ch. 3.1–3.3)



Some material adopted from notes by Charles R. Dyer, University of Wisconsin-Madison, with thanks

Dr. Cynthia Matuszek – CMSC 671 *Slides adapted with thanks from: Dr. Marie desJardin*

Bookkeeping

- TA Office hours: M 3-4, W 2-3
- General HW 1 questions?
- Basic Python
  - Sets, Tuples, Lists, Dictionaries, …
  - https://www.tutorialspoint.com/python
  - http://tiny.cc/concise-python-guide
  - http://www.w3resource.com/python/python-tutorial.php
  - https://docs.python.org/3
    - Especially Library Reference → Built-in Functions

We will put these on Piazza

Bits From Last Time

- **Sequential:** Requi[…] determine next be[…]
  - Or: current action […]
- **Episodic:** A series […]
  - Only the current pe[…]
  - Sensing/acting in e[…]
- **Single-** vs. **multi-**[…] one affecting the w[…]

Properties of the agent's *environment* (PEAS)

en.wikibooks.org/wiki/Artificial_Intelligence/AI_Agents_and_their_Environments
jeffclune.com/courses/media/courses/2014-Fall-AI/lectures/L04-AI-2014.pdf

What's a "State"?

- The current state of the agent's environment
- Everything in the **problem representation**
- Values of all parameters at a particular point in time
- Examples:
  - **Chess board:** 8x8 grid, location of all pieces
  - **Tic-tac-toe:** 3x3 grid, whether each is X, O, or open
  - **Robot soccer:** Location of all players, location of ball, *possibly* last known trajectory of all players (if sequential)
  - **Travel:** Cities, distances between cities, agent's current city

Some Examples

Agent Type	Performance Measure	Environment	Actuators	Sensors
Robot soccer player	Winning game, goals for/against	Field, ball, own team, other team, own body	Devices (e.g., legs) for locomotion and kicking	Camera, touch sensors, accelerometers, orientation sensors, wheel/joint encoders
Internet book-shopping agent	Obtain requested/ Interesting books, minimize expenditure	Internet	Follow link, enter/submit data in fields, display to user	Web pages, user requests

PEAS

Task Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Robot soccer	Partially	Stochastic	Sequential	Dynamic	Continuous	Multi
Internet book-shopping	Partially	Deterministic	Sequential	Static	Discrete	Single

Environment

Today's Class

- Goal-based agents
- Representing states and operators
- Example problems
- Generic state-space search algorithm

Everything in AI comes down to search.

Goal: understand search, and understand why.

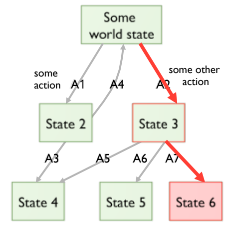
Pre-Reading Review

- What is **search** (a.k.a. **state-space search**)?
- What are these concepts in search?
  - Initial state
  - State space graph
  - Goal test (cf. goal)
  - Actions
  - Transition model
  - Step cost
  - Path cost
  - Solution / optimal solution
- What is an **open-loop system**?
- What is the difference between **expanding** and **generating** a state?
- What is the **frontier** (a.k.a. **open list**)?

7

Search: The Core Idea

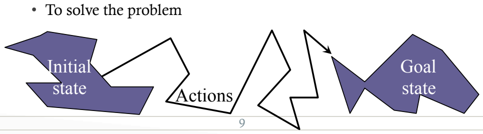
- For any problem:
  - World is (always) in some **state**
  - Agents take actions, which change the state
- We need a **sequence of actions** that gets the world into a particular **goal state**.
- To find it, we **search** the space of actions and states.



8

Building Goal-Based Agents

- To build a goal-based agent we need to decide:
  - What is the goal to be achieved?
  - What are the possible actions?
  - What **relevant** information must be encoded?
    - To describe the state of the world
    - To describe the available transitions
    - To solve the problem



9

What is the Goal?

- A situation we want to achieve
- A set of properties that we want to hold
- Must define a **"goal test"**
  - What does it mean to achieve it?
  - Have we done so?
- This is a hard question that is rarely tackled in AI!
  - Often, we assume the system designer or user will specify the goal
- For people, we stress the importance of establishing clear goals for as the first step towards solving a problem.
  - What are your goals?
  - What problem(s) are you trying to solve?

10

What Are Actions?

- **Primitive actions** or events:
  - Make changes in the world
  - In order to achieve a (sub)goal
  - Actions are also known as *operators* or *moves*
- Examples:

Low-level:
- Chess: "advance a pawn"
- Navigation: "take a step"
- Finance: "sell 10% of stock X"

High-level :
- Chess: "clear a path for a queen"
- Navigation: "go home"
- Finance: "sell best-return shares"

11

Actions and Determinism

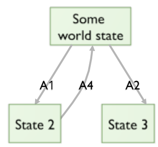
- In a **deterministic** world there is no uncertainty in an action's effects
- Current world **state** + chosen **action** fully specifies:

1. Whether that action can be done in current world
   - Is it applicable? (*E.g.*: Do I own any of stock X to sell?)
   - Is it legal? (*E.g.*: Can't just move a pawn sideways.)
2. World state after action is performed

12

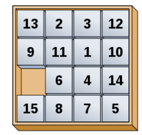
Representing Actions

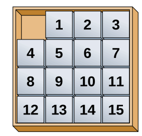
- Actions here are:
  - **Discrete events**
  - That occur at an **instant of time**
- For example:
  - State: “Mary is in class”
  - Action “Go home”
  - New state: “Mary is at home”
  - There is no representation of a state where she is in between (i.e., in the state of “going home”).



Sliding Tile Puzzles

- 15-puzzles, 8-puzzles
- How do we represent states?
- How do we represent actions?
  - Tile-1 moves north
  - Tile-1 moves west
  - Tile-1 moves east
  - Tile-1 moves south
  - Tile-2 moves north
  - Tile-2 moves west
  - …

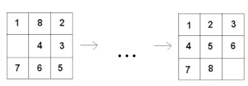




commons.wikimedia.org/wiki/File:15-puzzle-shuffled.svg, commons.wikimedia.org/wiki/File:15-puzzle-loyd-bis2.svg

Representing Actions

- Number of actions / operators depends on **representation** used in describing a state
- 8-puzzle:
  - Could specify 4 possible moves (actions) for each of the 8 tiles:
    4*8=32 operators.
  - Or, could specify four moves for the “blank” square:
    4 operators!



- **Careful representation can simplify a problem!**

Representing States

- What information about the world sufficiently describes all aspects **relevant** to solving the goal?
- That is: what knowledge must be in a state description to adequately describe the current state of the world?
- The **size of a problem** is usually described in terms of the **number of states** that are possible
  - Tic-Tac-Toe has about 3^9 states.
  - Checkers has about 10^{40} states.
  - Rubik’s Cube has about 10^{19} states.
  - Chess has about 10^{120} states in a typical game.

This is *ten quintillion* states.

Closed World Assumption

- We generally use the **Closed World Assumption**:

  “All necessary information about a problem domain is available in each percept so that each state is a complete description of the world.”

- No incomplete information at any point in time.
  - A statement that is true is always known to be true.
  - ∴ If we do not know something is true, it is false.

en.wikipedia.org/wiki/Closed-world_assumption

Some Example Problems

- Toy problems and micro-worlds
  - 8-Puzzle
  - Boat Problems
  - Cryptarithmetic
  - Remove 5 Sticks
  - Water Jug Problem
- Real-world problems

8-Puzzle

Given an initial configuration of 8 numbered tiles on a 3 x 3 board, move the tiles in such a way so as to produce a desired goal configuration of the tiles.

5	4	
6	1	8
7	3	2

Start State

1	2	3
8		4
7	6	5

Goal State

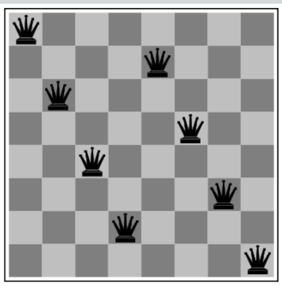
8-Puzzle

- **State:** 3 x 3 array describing where tiles are
- **Operators:** Move blank square Left, Right, Up or Down
  - This is a more efficient encoding of the operators!
- **Initial State:** Starting configuration of the board
- **Goal:** Some configuration of the board

5	4	
6	1	8
7	3	2

The 8-Queens Problem

Place eight (or N) queens on a chessboard such that no queen can reach any other



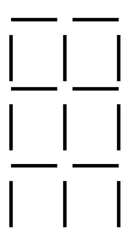
Boat Problems

1 sheep, 1 wolf, 1 cabbage, 1 boat

- **Goal**: Move everything across the river.
- **Constraints:**
  - The boat can hold you plus one thing.
  - Wolf can never be alone with sheep.
  - Sheep can never be alone with cabbage.
- **State:** location of sheep, wolf, cabbage on shores and boat.
- **Operators:** Move ferry containing some set of occupants across the river (in either direction) to the other side.

Remove 5 Sticks

- Given the following configuration of sticks, remove exactly 5 sticks in such a way that the remaining configuration forms exactly 3 squares.



Mathematical operations

- Proposed by Knuth (R&N p 73)
- Compute any positive integer, starting with the integer 4, using only factorial, square root, and floor operations
- Infinite state space!

$$\left\lfloor \sqrt{\sqrt{\sqrt{\sqrt{\sqrt{(4!)!}}}}} \right\rfloor = 5$$

- **States:** Positive numbers
- **Initial state:** 4
- **Actions:** Factorial (of integer states), square root, floor
- **Transition model:** Using mathematical definitions of actions
- **Goal test:** State is the desired positive integer

Some Real-World Problems

- Route finding
- Touring (traveling salesman)
- Logistics
- VLSI layout
- Robot navigation
- Learning

Knowledge Representation Issues

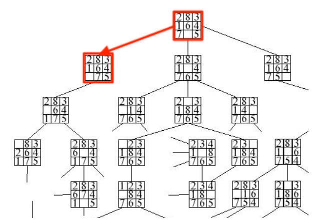
- What's in a state?
  - Is the color of the tiles relevant to solving an 8-puzzle?
  - Is sunspot activity relevant to predicting the stock market?
- What to represent is a very hard problem!
  - Usually left to the system designer to specify.
- What **level of abstraction** to describe the world?
  - Too fine-grained and we "miss the forest for the trees"
  - Too coarse-grained and we miss critical information

Knowledge Representation Issues

- Number of states depends on:
  - Representation choices
  - Level of abstraction
- In the Remove-5-Sticks problem:
  - If we represent individual sticks, then there are 17-choose-5 possible ways of removing 5 sticks (6188)
  - If we represent the "squares" defined by 4 sticks, there are 6 squares initially and we must remove 3
  - So, 6-choose-3 ways of removing 3 squares (20)

Formalizing Search in a State Space

- A *state space* is a **graph** (V, E):
  - V is a set of **nodes**
  - E is a set of **arcs**
  - Each arc is directed from a node to another node
- How does that work for 8-puzzle?



Formalizing Search in a State Space

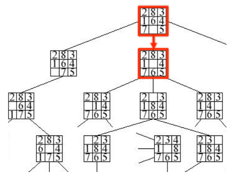
- **V**: A **node** is a data structure that contains:
  - State description
  - Bookkeeping information: parent(s) of the node, name of operator that generated the node from that parent, etc.
- **E**: Each **arc** is an instance (single occurrence) of one operator.
  - When operator is applied to the arc's source node (state), *then*
  - Resulting state is associated with the arc's destination node

Formalizing Search

- Each arc has a fixed, positive **cost**
  - Corresponding to the cost of the operator
  - What is "cost" of doing that action?
- Each node has a set of **successor nodes**
  - Corresponding to all operators (actions) that can apply at source node's state
  - **Expanding** a node is **generating** successor nodes, and adding them (and associated arcs) to the state-space graph

Formalizing Search II

- One or more nodes are designated as **start nodes**
- A **goal test** predicate is applied to a *state* to determine if its associated node is a goal node



Water Jug Problem

Given a full 5-gallon jug and an empty 2-gallon jug, the goal is to fill the 2-gallon jug with exactly one gallon of water.

State = (x,y), where x is the number of gallons of water in the 5-gallon jug and y is # of gallons in the 2-gallon jug

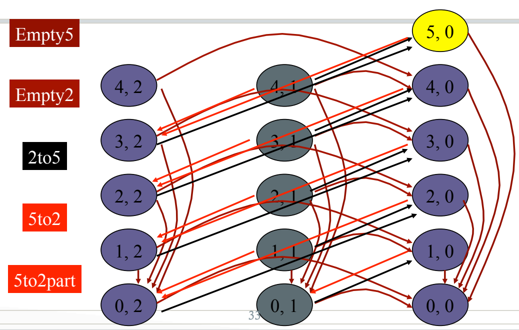
Initial State = (5,0)

Goal State = (*,1)
(* means any amount)

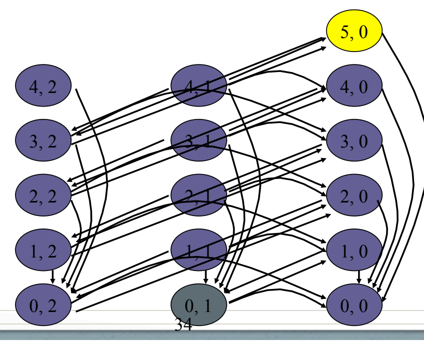
Operator table

Name	Cond.	Transition	Effect
Empty5	–	(x,y)→(0,y)	Empty 5-gal. jug
Empty2	–	(x,y)→(x,0)	Empty 2-gal. jug
2to5	x ≤ 3	(x,2)→(x+2,0)	Pour 2-gal. into 5-gal.
5to2	x ≥ 2	(x,0)→(x-2,2)	Pour 5-gal. into 2-gal.
5to2part	y < 2	(1,y)→(0,y+1)	Pour partial 5-gal. into 2-gal.

Water jug state space



Water jug solution



CLASS EXERCISE

- Representing a Sudoku puzzle as a search space
  - What are the states?
  - What are the operators?
  - What are the constraints (on operator application)?
  - What is the description of the goal state?
- Let's try it!

	3		
			1
3			
		2	

Formalizing Search III

- A **solution** is a sequence of operators that is associated with a path in a state space from a start node to a goal node.
- The **cost of a solution** is the sum of the arc costs on the solution path.
  - If all arcs have the same (unit) cost, then the solution cost is just the length of the solution (number of steps / state transitions)

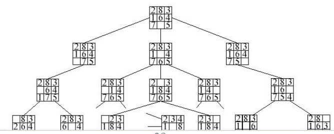
Formalizing Search IV

- **State-space search:** searching through a state space for a solution by **making explicit** a sufficient portion of an **implicit** state-space graph to find a goal node
  - Initially V={S}, where S is the start node
  - When S is **expanded**, its successors are **generated**; those nodes are added to V and the arcs are added to E
  - This process continues until a goal node is found
- It isn't usually practical to represent entire space

37

Formalizing Search V

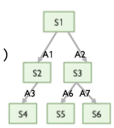
- Each node implicitly or explicitly represents a **partial solution path** (and its cost) from start node to given node.
  - In general, from a node there are many possible paths (and therefore solutions) that have this partial path as a prefix



38

State-Space Search Algorithm

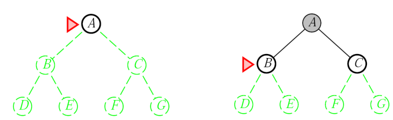
```
function general-search (problem, QUEUEING-FUNCTION)
;; problem describes start state, operators, goal test,
;;    and operator costs
;; queueing-function is a comparator function that
;;    ranks two states
;; returns either a goal node or failure

nodes = MAKE-QUEUE(MAKE-NODE(problem.INITIAL-STATE))
  loop
     if EMPTY(nodes) then return "failure"
     node = REMOVE-FRONT(nodes)
     if problem.GOAL-TEST(node.STATE) succeeds
        then return node
     nodes = QUEUEING-FUNCTION(nodes, EXPAND(node,
             problem.OPERATORS))
 end
     ;; Note: The goal test is NOT done when nodes are generated
     ;; Note: This algorithm does not detect loops
```



39

Generation vs. Expansion

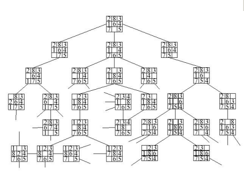
- **Selecting** a state means making that node current
- **Expanding** the current state means applying every legal action to the current state
- Which **generates** a new set of nodes



40 *R&N pg. 68, 80*

Key Procedures

- EXPAND
  - Generate **all** successor nodes of a given node
    - "What nodes can I reach from here (by taking what actions)?"
- GOAL-TEST
  - Test if state satisfies goal conditions
- QUEUEING-FUNCTION
  - Used to maintain a **ranked** list of nodes that are candidates for expansion
    - "What should I explore next?"



41

Algorithm Bookkeeping

- Typical node data structure includes:
  - State at this node
  - Parent node
  - Operator applied to get to this node
  - Depth of this node
    - That is, number of operator applications since initial state
  - Cost of the path
    - Sum of each operator application so far

42

Some Issues

- Search process constructs a search tree, where:
  - **Root** is the initial state and
  - **Leaf nodes** are nodes that are either:
    - Not yet expanded (i.e., they are in the list "nodes") or
    - Have no successors (i.e., they're "dead ends", because no operators can be applied, but they are not goals)
- Search tree may be infinite
  - Even for small search space
  - How?

Some Issues

- Return a path or a node depending on problem
  - In 8-queens return a **node**
  - 8-puzzle return a **path**
  - What about Sheep & Wolves?
- Changing definition of `Queueing-Function` → different search strategies
  - How do you choose what to expand next?

Evaluating Search Strategies

- Completeness:
  - Guarantees finding a solution if one exists
- Time complexity:
  - How long (worst or average case) does it take to find a solution?
  - Usually measured in number of states visited/nodes expanded
- Space complexity:
  - How much space is used by the algorithm?
  - Usually measured in maximum size of the "nodes" list during search
- Optimality / Admissibility
  - If a solution is found, is it guaranteed to be optimal (the solution with minimum cost)?