“A.I.” Faculty

- Murray – Automatic deduction and theorem proving. CSI 664
- Haas – Natural language processing, Prolog. CSI 636
- Davidson – Machine learning, probabilistic reasoning/learning, learning by compact encoding

Logistics and Next Lecture

- Text: Russell and Norvig – AI, 2nd Edition
- Mailing List, TA: ke@cs.albany.edu
- Meeting times: TuTh 10:15-11:35 @ ED0022
- Office hours: Tu/Th: 1pm -2pm
- Contact: davidson@cs.albany.edu, 518 442 5173
- 2 Assignments (25% each), Homeworks (10%), Final Exam (30%), Class participation (10%)
- Class web site: cs.albany.edu/~davidson, lectures
Chapter 2 – Russell and Norvig (not on exam)

Environment Properties - 1

- Variations in properties are why there are many A.I. techniques
- Fully versus partially observable
  - Agent perceives all information required to make the optimal action. Does not refer to violation of modeling assumptions.
  - Noisy sensors can make an environment partially observable
- Deterministic versus stochastic
  - From the view of the agent
  - Partially observable, stochastic environment = uncertainty
Environment Properties - 2

• Episodal versus Sequential
  – Most interesting environments are sequential
  – Each episode consists of the agent performing one action. If episodes are independent then episodal. ie $P^*(\text{Action}_t \mid P_t, P_{t-1}, P_1) = P^*(\text{Action}_t \mid P_t)$

Environment Properties - 3

• Static versus Dynamic
  – Semi-dynamic
• Discrete versus continuous
  – Applies to actions, states
• Single agent versus multiple agents
  – Co-operative and competitive agents
Name the Environment Properties

- Chess
- Monopoly
- Robocup simulation league
- Email crawler

Search – Chapter 3

Uninformed search

- Problem-solving agents
- Problem types
- Problem formulation
- Example problems
- Basic search algorithms
Goal Oriented Search – Ex: 1

On holiday in Romania; currently in Arad. Flight leaves tomorrow from Bucharest

Formulate goal:
be in Bucharest

Formulate problem:
states: various cities
operators: drive between cities

Find solution:
sequence of cities, e.g., Arad, Sibiu, Fagaras, Bucharest
Four Step Process

- Goal formulation, problem formulation, search, execute
- What does this tell you about the environment?
- How are the percepts for $t>1$ used?

Problem Types

**Deterministic, fully observable** $\Rightarrow$ single-state problem
Agent knows exactly which state it will be in; solution is a sequence

**Non-observable** $\Rightarrow$ conformant problem
Agent may have no idea where it is; solution (if any) is a sequence

**Nondeterministic and/or partially observable** $\Rightarrow$ contingency problem
Percepts provide new information about current state
Solution is a tree or policy
Often interleave search, execution

**Unknown state space** $\Rightarrow$ exploration problem (“online”)
Single State Problem Definition

A problem is defined by four items:

- **initial state**: e.g., "at Arad"
- **operators** (or successor function $S(x)$)
  - e.g., $\text{Arad} \rightarrow \text{Zerind}$
  - $\text{Arad} \rightarrow \text{Sibiu}$
  - etc.
- **goal test**, can be
  - explicit, e.g., $x = \text{"at Bucharest"}$
  - implicit, e.g., $\text{NoDirt}(x)$
- **path cost** (additive)
  - e.g., sum of distances, number of operators executed, etc.

A solution is a sequence of operators leading from the initial state to a goal state.

State Space Definition

Real world is absurdly complex

$\Rightarrow$ state space must be abstracted for problem solving

(Abstract) state = set of real states

(Abstract) operator = complex combination of real actions
  - e.g., "Arad $\rightarrow$ Zerind" represents a complex set of possible routes, detours, rest stops, etc.

For guaranteed realizability, any real state "in Arad" must get to some real state "in Zerind"

(Abstract) solution =
  - set of real paths that are solutions in the real world

Each abstract action should be "easier" than the original problem!
Eight State Puzzle

**States??**: integer locations of tiles (ignore intermediate positions)
**Actions??**: move blank left, right, up, down (ignore unjamming etc.)
**Goal test??**: = goal state (given)
**Path cost??**: 1 per move

[Note: optimal solution of n-Puzzle family is NP-hard]
Converting the Problem to a Graph

Basic Uninformed Search

Basic idea:
offline, simulated exploration of state space
by generating successors of already-explored states
(a.k.a. expanding states)

```
function GENERAL-SEARCH(problem, strategy) returns a solution, or failure
  initialize the search tree using the initial state of problem
  loop do
    if there are no candidates for expansion then return failure
    choose a leaf node for expansion according to strategy
    if the node contains a goal state then return the corresponding solution
    else expand the node and add the resulting nodes to the search tree
  end
```

CSI 535 - Introduction to A.I. Lecture 2+3
Uninformed Search
What is a State and a Node?

A state is a (representation of) a physical configuration.
A node is a data structure constituting part of a search tree
includes parent, children, depth, path cost \( g(n) \).

States do not have parents, children, depth, or path cost! Why?

The expand function creates new nodes, filling in the various fields and
using the operators (or successor function) of the problem to create the
 corresponding states.

Search Strategies and Effectiveness

A strategy is defined by picking the order of node expansion.

Strategies are evaluated along the following dimensions:
completeness—does it always find a solution if one exists?
time complexity—number of nodes generated/expanded
space complexity—maximum number of nodes in memory
optimality—does it always find a least-cost solution?

Time and space complexity are measured in terms of
\( b \)—maximum branching factor of the search tree
\( d \)—depth of the least-cost solution
\( m \)—maximum depth of the state space (may be \( \infty \))
Basic Search Algorithm

function TREE-Search(problem, fringe) returns a solution, or failure
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE(problem)), fringe)
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST(problem) applied to STATE(node) succeeds return node
        fringe ← INSERT-ALL(Expand(node, problem), fringe)
end loop

function Expand(node, problem) returns a set of nodes
    successors ← the empty set
    for each action, result in SUCCESSOR-FN(problem)(STATE(node)) do
        s ← a new Node
        PARENT-NODE(s) ← node; ACTION(s) ← action; STATE(s) ← result
        PATH-COST(s) ← PATH-COST(node) + STEP-COST(node, action, s)
        DEPTH(s) ← DEPTH(node) + 1
        add s to successors
    return successors

Uninformed Search

Uninformed strategies use only the information available in the problem definition

Breadth-first search
Uniform-cost search
Depth-first search
Depth-limited search
Iterative deepening search
Breadth First Search

Expand shallowest unexpanded node

**Implementation:**

What queue type should the fringe be?

- Complete?
- Time?
- Space?
- Optimal?

---

Uniform Search

Expand least-cost unexpanded node

**Implementation:**

Fringe queue order?

Equivalent to breadth-first if step costs all equal

- Complete?
- Time?
- Space?
- Optimal?
Depth First and Depth Limited

Expand deepest unexpanded node

Implementation:

Fringe queue order?

Complete?
Time?
Space?
Optimal?

Iterative Deepening and Bidirectional Search

Limit = 1
Limit = 2

Complete?
Time?
Space?
Optimal?
Formalizing the State Space - 1

- A **state space** is a graph, \((V, E)\), where \(V\) is a set of **nodes** and \(E\) is a set of **arcs**, where each arc is **directed** from a node to another node.
- Each **node** is a data structure, represents a particular state.
- Each **arc** corresponds to an instance of one of the operators. When the operator is applied to the state associated with the arc’s source node, then the resulting state is the state associated with the arc’s destination node.
- Each arc has a fixed, positive cost associated with it corresponding to the cost of the operator.

Formalizing the State Space - 2

- Each node has a set of **successor nodes** = all legal operators.
- 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**.
- 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.
- State-space search is searching through a state space for a solution by making explicit a sufficient portion of an implicit state-space graph to include a goal node. Hence, initially \(V=\{S\}\), where \(S\) is the start node; when \(S\) is expanded and so forth until …
- Each node implicitly or explicitly represents a **partial solution path** (and cost of the partial solution path) from the start node.
Basic Search Algorithm

```plaintext
function TREE-SEARCH(problem, fringe) returns a solution, or failure
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST(problem) applied to STATE(node) succeeds return node
        fringe ← INSERT-ALL(EXPAND(node, problem), fringe)
```

```plaintext
function EXPAND(node, problem) returns a set of nodes
    successors ← the empty set
    for each action, result in SUCCESSOR-FN(problem)(STATE(node)) do
        s ← a new NODE
        PARENT-NODE[s] ← node; ACTION[s] ← action; STATE[s] ← result
        PATH-COST[s] ← PATH-COST[node] + STEP-COST(node, action, s)
        DEPTH[s] ← DEPTH[node] + 1
        add s to successors
    return successors
```

Uninformed Search

*Uninformed* strategies use only the information available in the problem definition

- Breadth-first search
- Uniform-cost search
- Depth-first search
- Depth-limited search
- Iterative deepening search
Simple Example - 1

\[ S \ldots \text{Initial State} \]

- Depth-First Search: S A D E G
  - Solution found: S A G
- Breadth-First Search: S A B C D E G
  - Solution found: S A G
- Uniform-Cost Search: S A B D C E G
  - Solution found: S B G
  - This is the only uninformed search that worries about costs.
- Iterative-Deepening Search: S A B C S A D E G
  - Solution found: S A G

Nodes expanded by:

CSI 535 - Introduction to A.I. Lecture 2+3 - Uninformed Search

Simple Example - 2

\[ S \ldots \text{Initial State} \]

- Depth-first LIFO queue
  - expanded node
  - expanded nodes list

Uniform cost ordered queue

- Breadth-first FIFO queue
  - expanded node
  - expanded nodes list

CSI 535 - Introduction to A.I. Uninformed Search
Classic Problem #1: Eight Queen Problem

Classic Problem #2

9. Consider the classic farmer, fox, goose and grain problem. The farmer wants to move himself, the fox, the goose and the edible grain from the west to the east side of the river. Only he can row his small boat across the river and he can only take along with himself either the fox, the goose or the grain. That is, he can only take one of his items with him. If the fox is left with the goose, the goose will be eaten. If the goose is left with the grain, the grain will be eaten. You are to formally formulate this situation as a state space search problem. For example, you must state what variables comprise your state space.

a) Describe the representation of the state space and the goal test
b) Describe the operators that move from one state to another
c) Formally specify the constraints that are applicable for each operator
d) Describe a non-trivial admissible heuristic that is usable with the A* informed search algorithm
More Classic Problems

• Missionaries and Cannibals
  There are 3 missionaries, 3 cannibals, and 1 boat that can carry up to two people on one side of a river. Goal: Move all the missionaries and cannibals across the river. Constraint: Missionaries can never be outnumbered by cannibals on either side of the river, or else the missionaries are killed. State = configuration of missionaries and cannibals and boat on each side of the river. Operators: Move boat 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.

```
  -   -   -
  -   -   -
  -   -   -
```

• Water Jug Problem
  Given a 5-gallon jug and a 2-gallon jug, with the 5-gallon jug initially full of water and the 2-gallon jug empty, the goal is to fill the 2-gallon jug with exactly one gallon of water.

Water Jug Solution

- State = (x,y), where x = number of gallons of water in the 5-gallon jug and y in gallons in the 2-gallon jug.
- Initial State = (5,0)
- Goal State = (*,1), where * means any amount
- Operators
  - (x,y) -> (0,y), empty 5-gal jug
  - (x,y) -> (x,0), empty 2-gal jug
  - (x,2) and x<3 -> (x+2,0), pour 2-gal into 5-gal
  - (x,0) and x>=2 -> (x-2,2), pour 5-gal into 2-gal
  - (1,0) -> (0,1), empty 5-gal into 2-gal
- State Space (also called the Problem Space)

```
(5,0) - Start
  / \           
(3,2)  (0,6)  
  /   \       
(3,0)  (0,2)  
    /     
(1,2)  (0,1)  
    /     
(1,0)  
```

CSI 535 - Introduction to A.I. Lecture 2+3 - Uninformed Search
Uniform Cost Search

Next Class

- This class uninformed search
- Next classes are informed (heuristic) search (i.e. A*, Chapter 4, R&N)