Single Agent Search
COMP-4704-1

Prof. Nathan Sturtevant

Contact Info
- Office: 112 JGH
- email: sturtevant@cs.du.edu
- OH: TBA
- IM: nathansttt (yahoo, AIM), nathanst@gmail.com (MSN)

Syllabus Overview
- Prerequisites
- Cheating/Sharing of work
- Course assignments (Group component)
- May use chapters from unpublished textbook
- Motivation:
  - Excellent course project could become research paper (AIIDE)
### Coursework

<table>
<thead>
<tr>
<th>Coursework</th>
<th>Weighting</th>
<th>Due Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homework Assignments (3)</td>
<td>45%</td>
<td>Sept 30, Oct 21, Nov 19</td>
</tr>
<tr>
<td>Midterm</td>
<td>25%</td>
<td>Nov 9</td>
</tr>
<tr>
<td>Course Project</td>
<td>25%</td>
<td>Nov 19</td>
</tr>
<tr>
<td>Participation</td>
<td>5%</td>
<td></td>
</tr>
</tbody>
</table>

### Other details

- For some lectures will post readings ahead of time
- Mostly research papers
  - Practice reading and understanding technical writing
  - Observe and improve your own writing
  - Imitate in your homework

### Course Project

- Group project
  - Build test set of maps for AI community
  - Build set of test problems
  - Experiment with and categorize properties of maps
- Open to feedback

### Midterm

- After most lecture material has been presented
- To make sure you are attending class and paying attention
- Will note topics in lecture which may make good exam questions
Future possibilities

- If you need a reference letter in the future:
  - Get an A or A+ in the class
  - Do an excellent job working with and helping other students
  - Look for excellent students to work on interesting problems

Course Overview: Search Spaces

- Given:
  - Start state
  - Goal state (or states/goal test)
  - Operators

Course Overview: Search Spaces

- Find:
  - Path from start to goal
  - Set of states:
    - $S_1, S_2, \ldots S_i, S_{i+1}, \ldots S_N$
    - Where $S_{i+1} \in \text{successors}(S_i)$
  - Minimize cost of path

Course Overview: Search Spaces

- Successor function defines a graph
  - Nodes are states
  - Edge for each successor
  - Find path in graph from start to goal
  - All search algorithms can be seen as pathfinding in a graph
  - Search space can be explicit or implicit
  - Goal state can be explicit or implicit
Useful paradigm

- Often the state of a program is not easily encapsulated and/or testable
- If we can:
  - query the state for possible actions
  - apply and undo actions
- It is far easier to verify the correctness of an implementation

Example Domains

- Pathfinding
  - Commonly used in robotics, computer games, map applications
  - Find path on 2D map between start and goal
  - Potentially have multiple units and/or cooperation
  - Potential have additional dimensions
    - Speed, heading
Planning in Road Networks

- European graph has ~30 million nodes
- Find optimal path/length in <1ms
- Longer paths are easier!

Use memory to speed things up...

Example Domains

- Sliding Tile Puzzle
  - Simple to represent, difficult to solve
  - 8-puzzle has 9! = 362,880 states
  - 15-puzzle has 16! = $10^{13}$ states
    - Can be solved in less than 1 second
    - 36,000 nodes on average
  - Sam Loyd claimed to invent in the 1870’s (false)
  - $1000 cash prize for solving
Example Domains

• Rubik’s Cube
  • Invented in 1974 by Ernő Rubik
  • $10^{19}$ states
  • First solved optimally in 1997 (Korf)
  • Up to 17 CPU-days to solve one instance
    • 100 CPU-years if using previous techniques
  • $\leq 20$ moves to solve any position

Example Domains

• Rush Hour
  • Relatively easy for computer to solve

• Edit Distance
  • FOUR
  • FIVE

Example Domains

• Topspin Puzzle
• Flipside Puzzle
• The Missing Link

Multiple Sequence Alignment

• A set of $N$ sequences of DNA/RNA/protein
• A cost for matching, mismatching, blank
• Find optimal path through $n$-dimensional hypercube

```
PSHLQHYERTHGERPYECHQCGQAFKLCSLQHYERTHGERPYE-CNQCGQAFAQ-
SHLQHYERTHGERPYECHQCGQAFSQQHLLQHYERTHGERPYEMVIMAVKPLNES
```
Robotic Arm

- Move the tip of the arm to a desired location
- State is location of the arms

Rith

- Find equations in a grid of numbers
  - Cannot re-use numbers
  - Subsequent numbers must be adjacent
- How would we find solutions?
- How is this different?

CS (Graph Theory) vs. AI

- Graph theory also looks at explicit graphs
  - $|V|$ vertices
  - $|E|$ edges (At most $O(|V|^2)$ edges)
- AI often uses implicit graphs
  - Duplicate nodes hard to detect
- AI often assumes a constant branching factor
CS (Graph Theory) vs. AI

- CS considers time polynomial in (nodes + edges) efficient
- AI (usually) considers this too expensive
  - representation of implicit states takes $\log(n + e)$

CS (Graph Theory) vs. AI

- CS is willing to have everything stored in memory
  - polynomial in (nodes+edges)
- AI considers this exponential memory

But...

- Pathfinding is a special case
- Motivated by computer games
- Graph fits in memory
- Need extremely fast pathfinding
- Willing to give up optimality

Overview

- Problem Definition
- Assumptions
- Algorithms:
  - BFS
  - DFS
  - DFID
Overview

- Metrics
  - Complete / Optimal
  - Solution Quality
  - Time Complexity
  - Space Complexity
- Assumption
  - Uniform edges costs
  - Brute-force search, solution depth $d$

BFS - Time Complexity

$$N(b, d) = 1 + b + b^2 + \ldots + b^{d-1} + b^d$$

$$b \cdot N(b, d) = b + b^2 + \ldots + b^{d-1} + b^d + b^{d+1}$$

$$b \cdot N(b, d) - N(b, d) = b^{d+1} - 1$$

$$N(b, d) \cdot (b - 1) = b^{d+1} - 1$$

$$N(b, d) = \frac{b^{d+1} - 1}{b - 1}$$

$$N(b, d) \approx \frac{b^{d+1}}{b - 1} = b^d \frac{b}{b - 1}$$

BFS - Space Complexity

- Same as # nodes expanded $O(b^d)$
- Suppose:
  - 1 byte per node
  - 100 CPU cycles / node generation
  - 3Ghz machine
  - 30 million nodes (bytes) per second
  - 1 minute to fill ~2GB of memory