Problem Spaces & Search

CSE 473
473 Topics

Agents & Environments
Problem Spaces
Search & Constraint Satisfaction
Knowledge Repr’n & Logical Reasoning
Machine Learning
Uncertainty: Repr’n & Reasoning
Dynamic Bayesian Networks
Markov Decision Processes
Weak Methods

“In the knowledge lies the power…”
[Feigenbaum]

But what if no knowledge????
Generate & Test

As weak as it gets...

Works on semi-decidable problems!
Example:
Fragment of 8-Puzzle Problem Space
Search thru a Problem Space / State Space

Input:
- Set of states
- Operators [and costs]
- Start state
- Goal state [test]

Output:
- Path: start ⇒ a state satisfying goal test
- [May require shortest path]
Example: Route Planning

Input:

• Set of states

• Operators [and costs]

• Start state

• Goal state (test)

Output:
Example: N Queens

Input:
- Set of states
- Operators [and costs]
- Start state
- Goal state (test)

Output
Example: Blocks World

Input:

- Set of states
  Partially specified plans
- Operators [and costs]
  Plan modification operators
- Start state
  The null plan (no actions)
- Goal state (test)
  A plan which provably achieves
  The desired world configuration
Multiple Problem Spaces

Real World

States of the world (e.g. block configurations)
Actions (take one world-state to another)

Robot’s Head

**Problem Space 1**

PS states = models of world states
Operators = models of actions

**Problem Space 2**

PS states = partially spec. plan
Operators = plan modification ops
# Classifying Search

<table>
<thead>
<tr>
<th><strong>GUESSING</strong> (&quot;Tree Search&quot;)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Guess how to extend a partial solution to a problem.</td>
<td></td>
</tr>
<tr>
<td>• Generates a tree of (partial) solutions.</td>
<td></td>
</tr>
<tr>
<td>• The leafs of the tree are either “failures” or represent complete solutions</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>SIMPLIFYING</strong> (&quot;Inference&quot;)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Infer new, stronger constraints by combining one or more constraints (without any “guessing”)</td>
<td></td>
</tr>
<tr>
<td>Example: X+2Y = 3</td>
<td></td>
</tr>
<tr>
<td>X+Y = 1</td>
<td></td>
</tr>
<tr>
<td>therefore Y = 2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>WANDERING</strong> (&quot;Markov chain&quot;)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Perform a (biased) random walk through the space of (partial or total) solutions</td>
<td></td>
</tr>
</tbody>
</table>
Guessing – State Space Search
1. BFS
2. DFS
3. Iterative Deepening
4. Bidirectional
5. Best-first search
6. A*
7. Game tree
8. Davis-Putnam (logic)
9. Cutset conditioning (probability)

Simplification – Constraint Propagation
1. Forward Checking
2. Path Consistency (Waltz labeling, temporal algebra)
3. Resolution
4. “Bucket Algorithm”

Wandering – Randomized Search
1. Hillclimbing
2. Simulated annealing
3. Walksat
4. Monte-Carlo Methods
Search Strategies v2

Blind Search

- Depth first search
- Breadth first search
- Iterative deepening search
- Iterative broadening search

Informed Search

Constraint Satisfaction

Adversary Search
Depth First Search

Maintain stack of nodes to visit

Evaluation

- **Complete?**
  - Not for infinite spaces
- **Time Complexity?**
  - $O(b^d)$
- **Space Complexity?**
  - $O(d)$
Breadth First Search

Maintain queue of nodes to visit

Evaluation

• Complete?
  Yes

• Time Complexity?
  \( O(b^d) \)

• Space Complexity?
  \( O(b^d) \)
Memory a Limitation?

Suppose:
- 2 GHz CPU
- 1 GB main memory
- 100 instructions / expansion
- 5 bytes / node

200,000 expansions / sec
Memory filled in 100 sec ... < 2 minutes
Iterative Deepening Search

DFS with limit; incrementally grow limit

Evaluation

- **Complete?**
  Yes
- **Time Complexity?**
  $O(b^d)$
- **Space Complexity?**
  $O(d)$
## Cost of Iterative Deepening

<table>
<thead>
<tr>
<th>b</th>
<th>ratio ID to DFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1.5</td>
</tr>
<tr>
<td>10</td>
<td>1.2</td>
</tr>
<tr>
<td>25</td>
<td>1.08</td>
</tr>
<tr>
<td>100</td>
<td>1.02</td>
</tr>
</tbody>
</table>
When to Use Iterative Deepening

N Queens?
Search Space with Uniform Structure
Search Space with Clustered Structure
Iterative Broadening Search

What if know solutions lay at depth N?
No sense in doing iterative *deepening*

© Daniel S. Weld
Forwards vs. Backwards

Start

Arad

Zerind

Oradea

Sibiu

Fagaras

Neamt

Iasi

Vaslui

Timisoara

Lugoj

Mehadia

Dobreta

Craiova

Urziceni

Bucharest

Giurgiu

Hirsova

Eforie
vs. Bidirectional

Start

Goal
Problem

All these methods are slow (blind)

Solution → add guidance ("heuristic estimate")
→ “informed search”

... next time ...
Recap: Search thru a Problem Space / State Space

Input:
- Set of states
- Operators [and costs]
- Start state
- Goal state [test]

Output:
- Path: start ⇒ a state satisfying goal test
- [May require shortest path]
Cryptarithmetic

Input:
• Set of states

• Operators [and costs]

• Start state

• Goal state (test)

Output:

SEND
+ MORE
------
MONEY
Concept Learning

Labeled Training Examples

\(<p_1, \text{blond}, 32, \text{mc}, \text{ok}>\>
\(<p_2, \text{red}, 47, \text{visa}, \text{ok}>\>
\(<p_3, \text{blond}, 23, \text{cash}, \text{ter}>\>
\(<p_4, \ldots>\>

Output: \( f: <pn...> \rightarrow \{\text{ok, ter}\} \)

Input:

- Set of states
- Operators [and costs]
- Start state
- Goal state (test)

Output:
Symbolic Integration

E.g. \[ \int x^2 e^x \, dx = e^x(x^2 - 2x + 2) + C \]

 Operators:

Integration by parts
Integration by substitution

...