Artificial Intelligence
Strategy Making as Searching

Lecture 3
Week 3-4
Informed Search

• What happens if we have some idea (estimate) about “dilli kaha hai/kitna dur hai?”

• Can we mathematically model the estimation?

• How good is that estimation?
  • When can you say “it can’t go any worse”
  • When can it actually go worse than search without an estimate?
Heuristics

- A mathematical function that calculates the *estimate*

- **Input:**
  - Current state (what is the present situation)
  - All past states (what has already been explored)
  - Past experience (what life has taught the agent in similar situation)
  - Future prediction (what the agent *believes* about the world)

- **Output:**
  - Cost to goal state from current state for a particular action (*yet not taken*)
Heuristic for Pacman?
Manhattan Distance Vs. Euclidean Distance
Heuristic for Romania Problem?
Straight path heuristics
The Pancake Problem
Heuristic for Pancake Problem?
Heuristic: The largest pancake out
Who provides the heuristics?

- **Agent Developer**
  - By analyzing the characteristic of the problem
  - By fine-tuning past heuristics
  - By judging how good that heuristic is

- **Agent**
  - By coming up with a heuristic on its own
  - Developers gives a guideline of how to develop heuristic (*meta heuristics*)
When can a heuristic go wrong?

- Is it when the heuristic overestimates?
  - Calculates goal cost to be more than what it is

OR

- Is it when the heuristic underestimates?
  - Calculates goal cost to be less than what it is
Strategy 5: Best First Search

- Select the *most promising* node first
- How to *estimate* the *promise*?
  - What’s the current state?
  - What has the agent done so far?
  - What kind of goal the agent is working towards?
  - Is there any particular characteristic of the problem domain that helps?
- What kind of data structure has to be used?
What’s wrong with it?

- A common case:
  - Best-first takes you straight to the (wrong) goal

- Worst-case: like a badly-guided DFS in the worst case
  - Can explore everything
  - Can get stuck in loops if no cycle checking

- Like DFS in completeness (finite states w/ cycle checking)
Another case!
Strategy 6: A* = UCS + Best First Search

- Uniform-cost orders by path cost, or \( \text{backward cost} \ g(n) \)
- Greedy orders by goal proximity, or \( \text{forward cost} \ h(n) \)

- \( \text{A* Search} \) orders by the sum: \( f(n) = g(n) + h(n) \)
DO NOT grab the goal when you see it!

- Should we stop when we enqueue a goal?

- No: only stop when we dequeue a goal
Is A* Optimal?
What can go wrong?

- Heuristic function (estimate) was not good.
  - Estimate should not be pessimistic
  - Pessimistic estimate scares off the agent
  - That can actually be bad!

Scary estimate!
Friendly Heuristic (Admissible Heuristic)

- A heuristic $h$ is *admissible* (optimistic) if:

$$h(n) \leq h^*(n)$$

where $h^*(n)$ is the true cost to a nearest goal

- Examples:

- Coming up with admissible heuristics is most of what’s involved in using A* in practice.
A*: Proof of Optimality

Notation:
- $g(n) =$ cost to node $n$
- $h(n) =$ estimated cost from $n$ to the nearest goal (heuristic)
- $f(n) = g(n) + h(n) =$ estimated total cost via $n$
- $G^* =$ a lowest cost goal node
- $G =$ another goal node
Proof by contradiction

Proof:
- What could go wrong?
- We’d have to have to pop a suboptimal goal G off the fringe before G*
- This can’t happen:
  - Imagine a suboptimal goal G is on the queue
  - Some node n which is a subpath of G* must also be on the fringe (why?)
  - n will be popped before G

\[
\begin{align*}
f(n) &= g(n) + h(n) \\
g(n) + h(n) &\leq g(G^*) \\
g(G^*) &< g(G) \\
g(G) &= f(G) \\
f(n) &< f(G)
\end{align*}
\]
UCS Vs. A*
OK ... so what’s the big deal in it?

- A* is not a fast food algorithm
- Coming up with a *nice* and *friendly* heuristic function is challenging
- Is there a standard technique for that?
  - *Relaxation of the problem*
  - New types of imaginary actions are available (agents get high!)
    - Ex: Walking through a wall!
- NOTE: Scary heuristics are useful as well in some problems
  - When?
  - How?
8 Puzzle Problem

Start State

Goal State
8 Puzzle Problem: Solution Strategy

- What is a state for an agent?
  - How does the agent perceive this world?
- How many such states the agent can be in?
- What are all the different types of actions that an agent is armed with?
- What are all the possible states to be in given a current state?
- What is the penalization *(cost)* for a particular action that the agent takes?
  - How to formulate this penalization?
Heuristic for 8 Puzzle Problem?
Heuristic: Number of tiles misplaced

- Let’s call this strategy TILES
- Is this heuristic admissible (i.e. not scary)?

- What do we buy with TILES? \( h(\text{start}) = 8 \)
<table>
<thead>
<tr>
<th></th>
<th>4 steps</th>
<th>8 steps</th>
<th>12 steps</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average nodes</strong></td>
<td><strong>nodes</strong> expanded when optimal path has length...</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>UCS</strong></td>
<td>112</td>
<td>6,300</td>
<td>$3.6 \times 10^6$</td>
</tr>
<tr>
<td><strong>TILES</strong></td>
<td>13</td>
<td>39</td>
<td>227</td>
</tr>
</tbody>
</table>
Is there a better heuristic?
Heuristic: Manhattan distance

- Let’s call the strategy MANHATTAN
- Is this heuristic admissible (i.e. not scary)?

\[
h(\text{start}) = 18 \ (3+1+2+\ldots)
\]

- What do we buy with MANHATTAN?
Benefit of strategy MANHATTAN

<table>
<thead>
<tr>
<th></th>
<th>Average nodes expanded when optimal path has length…</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>…4 steps</td>
</tr>
<tr>
<td>TILES</td>
<td>13</td>
</tr>
<tr>
<td>MANHATTAN</td>
<td>12</td>
</tr>
</tbody>
</table>
What if Actual Cost easy to calculate?
The A* Trade-off

- Do you care about the quality of estimate?

OR

- Do you care about the work that an agent has to do per state?
Dominance: The Heuristics Fight Club!

- Dominance: $h_a \geq h_c$ if
  $$\forall n : h_a(n) \geq h_c(n)$$

- Heuristics form a semi-lattice:
  - Max of admissible heuristics is admissible
    $$h(n) = \max(h_a(n), h_b(n))$$

- Trivial heuristics
  - Bottom of lattice is the zero heuristic (what does this give us?)
  - Top of lattice is the exact heuristic
Where can we apply A*?

- Routing problems
- Pathing problems
- Resource planning problems
- Robot motion problems
- Natural Language analysis
- Machine Translation
- Speech Recognition
- And lot more! (you just need to creative to convert a problem into a single agent optimal goal search problem)
What’s wrong here?
The problem of repetition: BFS

- **DO NOT** expand the pink nodes

- **Never expand a state twice**
How do we implement it?
What happens if A* search is on graph?
What is going awry here?
Friendly is not enough (if you do it the graph way)!

- A heuristic function must be friendly at a particular situation (admissible at a particular node state)
- A heuristic function must also be friendly throughout a path to a particular situation (consistent at a particular)

```
A
  \[ h=4 \]
  \[ \text{edge to C with cost 1} \]

C
  \[ h=1 \]

G
  \[ \text{edge to C with cost 3} \]

\text{cost(A to C) + h(C) \geq h(A)}
\text{cost(A to C) \geq h(A) - h(C)}
\text{real cost \geq cost implied by heuristic}
```