Informed Search I

Outline for today’s lecture

Uninformed Search
• Briefly: Bidirectional Search (AIMA 3.4.6)
• “Uniform Cost” Search (UCS)

Informed Search
• Introduction to Informed search
  • Heuristics
• 1st attempt: Greedy Best-first search

Very briefly: Bidirectional search

• Two simultaneous searches from start and goal.
  • Motivation: In general, \[ h^2 + g^2 < h^2 \]
• Check whether the node belongs to the other frontier before expansion.
• Complete and optimal if both searches are Breadth-First.
• Most significant weakness: Space complexity

One issue: How to search backwards?

• The predecessor of each node must be efficiently computable.
  • Works well when actions are easily reversible.

“Uniform Cost” Search

“In computer science, uniform-cost search (UCS) is a tree search algorithm used for traversing or searching a weighted tree, tree structure, or graph.” - Wikipedia
Motivation: Romanian Map Problem

- All our search methods so far assume step-cost = 1
- This is only true for some problems

\[ g(N) : \text{the path cost function} \]

- Our assumption so far: All moves equal in cost
  - Cost = # of nodes in path-1
  - \( g(N) = \text{depth}(N) \) in the search tree

- More general: Assigning a (potentially) unique cost to each step
  - \( N_0, N_1, N_2, N_3 \) - nodes visited on path \( p \) from \( N_0 \) to \( N_3 \)
  - \( C(i,j) \): Cost of going from \( N_i \) to \( N_j \)
  - If \( N_0 \) the root of the search tree,
    \[ g(N_3) = C(0,1)+C(1,2)+C(2,3) \]

Uniform-cost search (UCS)

- Extension of BF-search:
  - Expand node with lowest path cost
- Implementation:
  - frontier = priority queue ordered by \( g(n) \)
- Subtle but significant difference from BFS:
  - Tests if a node is a goal state when it is selected for expansion, not when it is added to the frontier.
  - Updates a node on the frontier if a better path to the same state is found.
  - So always enqueues a node before checking whether it is a goal.

WHY???

Uniform Cost Search

Expand cheapest node first:
Frontier is a priority queue
No longer ply at a time, but follows cost contours
Therefore: Must be optimal

Complexity of UCS

- Complete!
- Optimal!
  - if the cost of each step exceeds some positive bound \( \epsilon \).
  - Time complexity: \( O(b^{C^*/\epsilon}+1) \)
  - Space complexity: \( O(b^{C^*/\epsilon}+1) \)
    where \( C^* \) is the cost of an optimal solution, and \( \epsilon \) is \( \min(C(i,j)) \)
    (if all step costs are equal, this becomes \( O(b^{C^*/\epsilon}) \))

NOTE: Dijkstra’s algorithm just UCS without goal

Summary of algorithms (for notes)

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Breadth-First</th>
<th>Uniform-cost</th>
<th>Depth-limited</th>
<th>Iterative deepening</th>
<th>Bidirectional search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete?</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>Time</td>
<td>( b^d )</td>
<td>( b^{C^*/\epsilon}+1 )</td>
<td>( b^d )</td>
<td>( b^d )</td>
<td>( b^{d/2} )</td>
</tr>
<tr>
<td>Space</td>
<td>( b^d )</td>
<td>( b^{C^*/\epsilon}+1 )</td>
<td>( b^d )</td>
<td>( b^d )</td>
<td>( b^{d/2} )</td>
</tr>
<tr>
<td>Optimal?</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
</tbody>
</table>

Assumes \( b \) is finite
Outline for today’s lecture

Uninformed Search
- Briefly: Bidirectional Search
- "Uniform Cost" Search (UCS)

Informed Search
- Introduction to Informed search
  - Heuristics
  - 1st attempt: Greedy Best-first search

Is Uniform Cost Search the best we can do?
Consider finding a route from Bucharest to Arad.

Is Uniform Cost Search the best we can do?
Consider finding a route from Bucharest to Arad...

A Better Idea...

- Node expansion based on an estimate which includes distance to the goal

- General approach of informed search:
  - Best-first search: node selected for expansion based on an evaluation function \( f(n) \)
     - \( f(n) \) includes estimate of distance to goal (new idea?)

- Implementation: Sort frontier queue by this new \( f(n) \).
  - Special cases: greedy search, \( A^\ast \) search

Simple, useful estimate heuristic: straight-line distances

Heuristic (estimate) functions

Heureka! --- Archimedes

[definition] “A rule of thumb, simplification, or educated guess that reduces or limits the search for solutions in domains that are difficult and poorly understood.”

Heuristic knowledge is useful, but not necessarily correct.

Heuristic algorithms use heuristic knowledge to solve a problem.

A heuristic function \( h(n) \) takes a state \( n \) and returns an estimate of the distance from \( n \) to the goal.

(graphic: http://hyperbolegames.com/2014/10/20/eureka-moments/)
Informed search (A*) in a world with obstacles

Outline for today's lecture

Uninformed Search
- Briefly: Bidirectional Search
- “Uniform Cost” Search (UCS)

Informed Search
- Introduction to Informed search
  - Heuristics
- 1st attempt: Greedy Best-first search (AIMA 3.5.1)
Review: Best-first search

Basic idea:
- select node for expansion with minimal evaluation function \( f(n) \)
  - where \( f(n) \) is some function that includes estimate heuristic \( h(n) \) of the remaining distance to goal
- Implement using priority queue
- Exactly UCS with \( f(n) \) replacing \( g(n) \)

Greedy best-first search: \( f(n) = h(n) \)

- Expands the node that is estimated to be closest to goal
- Completely ignores \( g(n) \): the cost to get to \( n \)
- Here, \( h(n) = h_{SLD}(n) \) = straight-line distance from \( n \) to Bucharest

Greedy best-first search example

Initial State = Arad
Goal State = Bucharest
Frontier queue:
Arad 366
Bucharest 0
Cluj-Napoca 160
Drobeta 242
Eforie 161
Fagaras 176
Giurgiu 77
Hunedoa 153
Iasi 254
Lugoj 244
Mehadia 244
Neamt 254
Oradea 380
Ploiesti 100
Rimnicu Vilcea 193
Sibiu 234
Timisoara 329
Zarand 374

Greedy best-first search example

Frontier queue:
Sibiu 253
Timisoara 329
Zarand 374
Arad 366
Bucharest 0
Cluj-Napoca 160
Drobeta 242
Eforie 161
Fagaras 176
Giurgiu 77
Hunedoa 153
Iasi 236
Lugoj 244
Mehadia 244
Neamt 254
Oradea 380
Ploiesti 100
Rimnicu Vilcea 193
Sibiu 234
Timisoara 329
Zarand 374
Oradea 380

Greedy best-first search example

Goal reached !!

Frontier queue:
Bucharest 0
Rimnicu Vilcea 193
Sibiu 253
Timisoara 329
Arad 366
Bucharest 0
Cluj-Napoca 160
Drobeta 242
Eforie 161
Fagaras 176
Giurgiu 77
Hunedoa 153
Iasi 236
Lugoj 244
Mehadia 244
Neamt 254
Oradea 380
Ploiesti 100
Rimnicu Vilcea 193
Sibiu 234
Timisoara 329
Zarand 374
Oradea 380

**Properties of greedy best-first search**

- **Optimal?**
  - No!
  - Found: Arad \(\rightarrow\) Sibiu \(\rightarrow\) Fagaras \(\rightarrow\) Bucharest (450km)
  - Shorter: Arad \(\rightarrow\) Sibiu \(\rightarrow\) Rimnicu Vilcea \(\rightarrow\) Pitesti \(\rightarrow\) Bucharest (418km)

- **Complete?**
  - No – can get stuck in loops,
    - e.g., Iasi \(\rightarrow\) Neamt \(\rightarrow\) Iasi \(\rightarrow\) Neamt \(\rightarrow\) ...

- **Time?** \(O(b^m)\) – worst case (like Depth First Search)
  - But a good heuristic can give dramatic improvement of average cost

- **Space?** \(O(b^m)\) – priority queue, so worst case: keeps all (unexpanded) nodes in memory

- **Optimal?** No

**Next time: A* search**

- **Best-known form of best-first search.**
- **Key Idea:** avoid expanding paths that are already expensive, but expand most promising first.

  - **Simple idea:** \(f(n) = g(n) + h(n)\)
    - \(g(n)\) the cost (so far) to reach the node
    - \(h(n)\) estimated cost to get from the node to the goal
    - \(f(n)\) estimated total cost of path through \(n\) to goal

- **Implementation:** Frontier queue as priority queue by increasing \(f(n)\) (as expected…)

**Optimal informed search: A**

- **Best-known informed search method**
- **Key Idea:** avoid expanding paths that are already expensive, but expand most promising first.

  - **Simple idea:** \(f(n) = g(n) + h(n)\)

- **Implementation:** Same as before
  - Frontier queue as priority queue by increasing \(f(n)\)
Key concept: Admissible heuristics

- A heuristic $h(n)$ is **admissible** if it never overestimates the cost to reach the goal; i.e. it is optimistic
  - Formally: for any node $n$:
    1. $h(n) \leq h^*(n)$ where $h^*(n)$ is the true cost from $n$
    2. $h(n) \geq 0$ so $h(G)=0$ for any goal $G$.

- **Example:** $h_{sld}(n)$ never overestimates the actual road distance

**Theorem:** If $h(n)$ is admissible, A* using Tree Search is **optimal**
A* search example

Frontier queue:
- Bucharest 418
- Timisoara 447
- Zerind 449
- Bucharest 450
- Craiova 526
- Sibiu 553
- Sibiu 591
- Suceava 607
- Craiova 615
- Arad 646
- Oradea 671

Note that we just found a better value for Bucharest!
Now we expand this better value for Bucharest since it's at the top of the queue.
We're done and we know the value found is optimal!