Search with Costs and Heuristic Search

CPSC 322 - Search 3
January 17, 2011

Textbook §3.5.3, 3.6.1
Taught by: Mike Chiang
Today’s Lecture

Recap from last lecture, combined with Alspace demo

• Search with costs: Least Cost First Search
• Heuristics Search: Best First Search
• Apply basic properties of search algorithms:
  - completeness
  - optimality
  - time and space complexity of search algorithms

• Select the most appropriate search algorithms for specific problems.
  – Depth-First Search vs. Breadth-First Search
DFS and BFS

Depth-First Search, DFS

Breadth-First Search, BFS

Let’s look at these algorithms in action:
Comparing Searching Algorithms: Will it find a solution? The best one?

Def.: A search algorithm is **complete** if whenever there is at least one solution, the algorithm is guaranteed to find it within a finite amount of time.

- BFS is complete, DFS is not

Def.: A search algorithm is **optimal** if when it finds a solution, it is the best one.

- BFS is optimal, DFS is not
Comparing Searching Algorithms: Complexity

**Def.:** The *time complexity* of a search algorithm is the worst-case amount of time it will take to run, expressed in terms of
- maximum path length $m$
- maximum forward branching factor $b$.

- Both BFS and DFS take time $O(b^m)$ in the worst case

**Def.:** The *space complexity* of a search algorithm is the worst-case amount of memory that the algorithm will use (i.e., the maximal number of nodes on the frontier).

- **BFS:** $O(b^m)$, $O(m^b)$, $O(bm)$, $O(b+m)$
- **DFS:** $O(b^m)$, $O(m^b)$, $O(bm)$, $O(b+m)$
Today’s Lecture

• Recap from last lecture, combined with Alspace demo
• Search with costs: Least Cost First Search
• Heuristic Search: Best First Search
Example: edge costs in the delivery robot domain
Search with Costs

• Sometimes there are costs associated with arcs.

Def.: The cost of a path is the sum of the costs of its arcs

\[ \text{cost}(\langle n_0, \ldots, n_k \rangle) = \sum_{i=1}^{k} \text{cost}(\langle n_{i-1}, n_i \rangle) \]

• In this setting we often don't just want to find any solution
  – we usually want to find the solution that minimizes cost

Def.: A search algorithm is optimal if
  when it finds a solution, it is the best one
Search with Costs

• Sometimes there are costs associated with arcs.

Def.: The cost of a path is the sum of the costs of its arcs

\[
\text{cost}(\langle n_0, \ldots, n_k \rangle) = \sum_{i=1}^{k} \text{cost}(\langle n_{i-1}, n_i \rangle)
\]

• In this setting we often don't just want to find any solution
  – we usually want to find the solution that minimizes cost

Def.: A search algorithm is optimal if
  when it finds a solution, it is the best one:
  it has the lowest path cost
Lowest-Cost-First Search (LCFS)

- **Lowest-cost-first search** finds the path with the lowest cost to a goal node.
- At each stage, it selects the path with the lowest cost on the frontier.
- The frontier is implemented as a priority queue ordered by path cost.

Let’s look at this in action:
When arc costs are equal, LCFS is equivalent to...

- DFS
- BFS
- IDS
- None of the above
Analysis of Lowest-Cost First Search

• Is LCFS complete?
  - Not in general: a cycle with zero cost, or negative arc costs could be followed forever
  - Yes, as long as arc costs are strictly positive

• Is LCFS optimal?
  
  YES  NO  IT DEPENDS

  - Not in general: arc costs could be negative: a path that initially looks high-cost could end up getting a "refund".
  - Yes, as long as arc costs are guaranteed to be non-negative.
Analysis of Lowest-Cost First Search

• What is the time complexity of LCFS if the maximum path length is \( m \) and the maximum branching factor is \( b \)?

\[
O(b^m) \quad O(m^b) \quad O(bm) \quad O(b+m)
\]

- Knowing costs doesn't help here; worst case: all nodes

• What is the space complexity?

\[
O(b^m) \quad O(m^b) \quad O(bm) \quad O(b+m)
\]

E.g. uniform cost: just like BFS, in worst case frontier has to store all nodes \( m-1 \) steps from the start node
“Uninformed Search”: DFS, BFS, LCFS

• Why are all these strategies called uninformed?
  - Because they do not consider any information about the states and the goals to decide which path to expand first on the frontier
  - They are blind to the goal

• In other words, they are general and do not take into account the specific nature of the problem.
Today’s Lecture

• Recap from last lecture, combined with Alspace demo
• Search with costs: Least Cost First Search

Heuristic Search: Best First Search
Blind search algorithms do not take into account the goal until they are at a goal node.

Often there is extra knowledge that can be used to guide the search:
- an estimate of the distance from node $n$ to a goal node.

This estimate is called search heuristic.
More formally

Def.: A search heuristic $h(n)$ is an estimate of the cost of the optimal (cheapest) path from node $n$ to a goal node.
Example: finding routes

- What could we use as $h(n)$?
Example: finding routes

- What could we use as \( h(n) \)? E.g., the straight-line (Euclidian) distance between source and goal node.
Best First Search (BestFS)

- Idea: always choose the path on the frontier with the smallest $h$ value.
- BestFS treats the frontier as a priority queue ordered by $h$.
- **Greedy** approach: expand path whose last node seems closest to the goal

Let’s look at this in action:  

Optimal? AISPACE example, load from URL  
• A low heuristic value can mean that a cycle gets followed forever -> not complete
Analysis of BestFS

- Complete? No, see the example last slide

- Optimal? No, see the Alspace example from above:

- Time Complexity
  \[ O(b^m) \quad O(m^b) \quad O(bm) \quad O(b+m) \]
  - Worst case: has to explore all nodes

- Space Complexity
  \[ O(b^m) \quad O(m^b) \quad O(bm) \quad O(b+m) \]
  - Heuristic could be such to emulate BFS:
    E.g. \( h(n) = (m - \text{distance of } n \text{ from start}) \)
Learning Goals for today’s class

• Select the most appropriate algorithms for specific problems.
  – Depth-First Search vs. Breadth-First Search vs. Least-Cost-First Search vs. Best-First Search

• Define/read/write/trace/debug different search algorithms

• Construct heuristic functions for specific search problems (just started, more on this next time)