Heuristic Search

CPSC 322 Lecture 6

September 17, 2007
Textbook §3.5

Lecture Overview

1. Recap
2. Breadth-First Search
3. Search with Costs

Graph Search Algorithm

Input: a graph,
a set of start nodes,
Boolean procedure $goal(n)$ that tests if n is a goal node.

$frontier := \{\langle s \rangle : s$ is a start node$\}$;
while $frontier$ is not empty:
  select and **remove** path $\langle n_0, \ldots, n_k \rangle$ from $frontier$;
  if $goal(n_k)$
    return $\langle n_0, \ldots, n_k \rangle$;
  for every neighbor n of n_k
    add $\langle n_0, \ldots, n_k, n \rangle$ to $frontier$;
end while

- After the algorithm returns, it can be asked for more answers and the procedure continues.
- Which value is selected from the frontier defines the search strategy.
- The *neighbor* relationship defines the graph.
- The *goal* function defines what is a solution.

Depth-first Search

- Depth-first search treats the frontier as a stack
  - It always selects one of the last elements added to the frontier.

- Complete when the graph has no cycles and is finite
- Time complexity is $O(b^m)$
- Space complexity is $O(bm)$

Using Depth-First Search

- When is DFS appropriate?

Using Depth-First Search

- When is DFS appropriate?
  - space is restricted
  - solutions tend to occur at the same depth in the tree
  - you know how to order nodes in the list of neighbours so that solutions will be found relatively quickly

Using Depth-First Search

- When is DFS appropriate?
  - space is restricted
  - solutions tend to occur at the same depth in the tree
  - you know how to order nodes in the list of neighbours so that solutions will be found relatively quickly

- When is DFS inappropriate?

Using Depth-First Search

- When is DFS appropriate?
    - space is restricted
    - solutions tend to occur at the same depth in the tree
    - you know how to order nodes in the list of neighbours so that solutions will be found relatively quickly

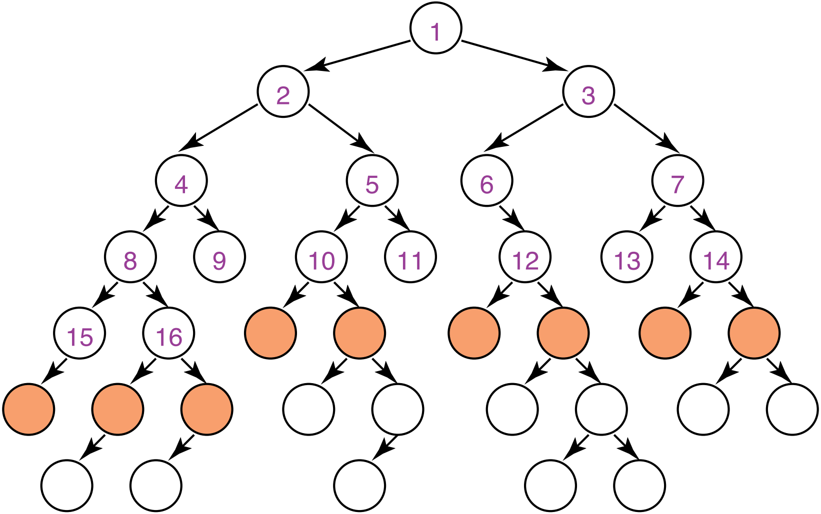
- When is DFS inappropriate?
    - some paths have infinite length
    - the graph contains cycles
    - some solutions are very deep, while others are very shallow

Lecture Overview

Breadth-first Search

- Breadth-first search treats the frontier as a queue
  - it always selects one of the earliest elements added to the frontier.
- Example:
  - the frontier is $[p_1, p_2, \ldots, p_r]$
  - neighbours of p_1 are $\{n_1, \ldots, n_k\}$
- What happens?
  - p_1 is selected, and tested for being a goal.
  - Neighbours of p_1 follow p_r at the end of the frontier.
  - Thus, the frontier is now $[p_2, \ldots, p_r, (p_1, n_1), \ldots, (p_1, n_k)]$.
  - p_2 is selected next.

Illustrative Graph — Breadth-first Search



Analysis of Breadth-First Search

- Is BFS complete?

Analysis of Breadth-First Search

- Is BFS complete?
  - Yes (but it wouldn't be if the branching factor for any node was infinite)
  - In fact, BFS is guaranteed to find the path that involves the fewest arcs (why?)

Analysis of Breadth-First Search

- Is BFS complete?
  - Yes (but it wouldn't be if the branching factor for any node was infinite)
  - In fact, BFS is guaranteed to find the path that involves the fewest arcs (why?)
- What is the time complexity, if the maximum path length is m and the maximum branching factor is b?

Analysis of Breadth-First Search

- Is BFS complete?
  - Yes (but it wouldn't be if the branching factor for any node was infinite)
  - In fact, BFS is guaranteed to find the path that involves the fewest arcs (why?)
- What is the time complexity, if the maximum path length is m and the maximum branching factor is b?
  - The time complexity is $O(b^m)$: must examine every node in the tree.
  - The order in which we examine nodes (BFS or DFS) makes no difference to the worst case: search is unconstrained by the goal.

Analysis of Breadth-First Search

- Is BFS complete?
  - Yes (but it wouldn't be if the branching factor for any node was infinite)
  - In fact, BFS is guaranteed to find the path that involves the fewest arcs (why?)
- What is the time complexity, if the maximum path length is m and the maximum branching factor is b?
  - The time complexity is $O(b^m)$: must examine every node in the tree.
  - The order in which we examine nodes (BFS or DFS) makes no difference to the worst case: search is unconstrained by the goal.
- What is the space complexity?

Analysis of Breadth-First Search

- Is BFS complete?
  - Yes (but it wouldn't be if the branching factor for any node was infinite)
  - In fact, BFS is guaranteed to find the path that involves the fewest arcs (why?)
- What is the time complexity, if the maximum path length is m and the maximum branching factor is b?
  - The time complexity is $O(b^m)$: must examine every node in the tree.
  - The order in which we examine nodes (BFS or DFS) makes no difference to the worst case: search is unconstrained by the goal.
- What is the space complexity?
  - Space complexity is $O(b^m)$: we must store the whole frontier in memory

Using Breadth-First Search

- When is BFS appropriate?

Using Breadth-First Search

- When is BFS appropriate?
  - space is not a problem
  - it's necessary to find the solution with the fewest arcs
  - although all solutions may not be shallow, at least some are
  - there may be infinite paths

Using Breadth-First Search

- When is BFS appropriate?
    - space is not a problem
    - it's necessary to find the solution with the fewest arcs
    - although all solutions may not be shallow, at least some are
    - there may be infinite paths

- When is BFS inappropriate?

Using Breadth-First Search

- When is BFS appropriate?
  - space is not a problem
  - it's necessary to find the solution with the fewest arcs
  - although all solutions may not be shallow, at least some are
  - there may be infinite paths
- When is BFS inappropriate?
  - space is limited
  - all solutions tend to be located deep in the tree
  - the branching factor is very large

Lecture Overview

1. Recap
2. Breadth-First Search
3. Search with Costs

Search with Costs

Sometimes there are costs associated with arcs.

Definition (cost of a path)

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

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

Search with Costs

Sometimes there are costs associated with arcs.

Definition (cost of a path)

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

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

In this setting we often don't just want to find just any solution

- we usually want to find the solution that minimizes cost

Definition (optimal algorithm)

A search algorithm is optimal if it is complete, and only returns cost-minimizing solutions.

Lowest-Cost-First Search

- At each stage, lowest-cost-first search selects a path on the frontier with lowest cost.
  - The frontier is a priority queue ordered by path cost.
  - We say "a path" because there may be ties
- When all arc costs are equal, LCFS is equivalent to BFS.

Lowest-Cost-First Search

- At each stage, lowest-cost-first search selects a path on the frontier with lowest cost.
  - The frontier is a priority queue ordered by path cost.
  - We say "a path" because there may be ties
- When all arc costs are equal, LCFS is equivalent to BFS.
- Example:
  - the frontier is $[\langle p_1, 10\rangle, \langle p_2, 5\rangle, \langle p_3, 7\rangle]$
  - p_2 is the lowest-cost node in the frontier
  - neighbours of p_2 are $\{\langle p_9, 12\rangle, \langle p_{10}, 15\rangle\}$
- What happens?

Lowest-Cost-First Search

- At each stage, lowest-cost-first search selects a path on the frontier with lowest cost.
  - The frontier is a priority queue ordered by path cost.
  - We say "a path" because there may be ties
- When all arc costs are equal, LCFS is equivalent to BFS.
- Example:
  - the frontier is $[\langle p_1, 10\rangle, \langle p_2, 5\rangle, \langle p_3, 7\rangle]$
  - p_2 is the lowest-cost node in the frontier
  - neighbours of p_2 are $\{\langle p_9, 12\rangle, \langle p_{10}, 15\rangle\}$
- What happens?
  - p_2 is selected, and tested for being a goal.
  - Neighbours of p_2 are inserted into the frontier (it doesn't matter where they go)
  - Thus, the frontier is now $[\langle p_1, 10\rangle, \langle p_9, 12\rangle, \langle p_{10}, 15\rangle, \langle p_3, 7\rangle]$.
  - p_3 is selected next.
  - Of course, we'd really implement this as a priority queue.

Analysis of Lowest-Cost-First Search

- Is LCFS complete?

Analysis of Lowest-Cost-First Search

- Is LCFS complete?
  - not in general: a cycle with zero or negative arc costs could be followed forever.

Analysis of Lowest-Cost-First Search

- Is LCFS complete?
  - not in general: a cycle with zero or negative arc costs could be followed forever.
  - yes, as long as arc costs are strictly positive
- What is the time complexity, if the maximum path length is m and the maximum branching factor is b?

Analysis of Lowest-Cost-First Search

- Is LCFS complete?
  - not in general: a cycle with zero or negative arc costs could be followed forever.
  - yes, as long as arc costs are strictly positive
- What is the time complexity, if the maximum path length is m and the maximum branching factor is b?
  - The time complexity is $O(b^m)$: must examine every node in the tree.
  - Knowing costs doesn't help here.

Analysis of Lowest-Cost-First Search

- Is LCFS complete?
  - not in general: a cycle with zero or negative arc costs could be followed forever.
  - yes, as long as arc costs are strictly positive
- What is the time complexity, if the maximum path length is m and the maximum branching factor is b?
  - The time complexity is $O(b^m)$: must examine every node in the tree.
  - Knowing costs doesn't help here.
- What is the space complexity?

Analysis of Lowest-Cost-First Search

- Is LCFS complete?
  - not in general: a cycle with zero or negative arc costs could be followed forever.
  - yes, as long as arc costs are strictly positive
- What is the time complexity, if the maximum path length is m and the maximum branching factor is b?
  - The time complexity is $O(b^m)$: must examine every node in the tree.
  - Knowing costs doesn't help here.
- What is the space complexity?
  - Space complexity is $O(b^m)$: we must store the whole frontier in memory.

Analysis of Lowest-Cost-First Search

- Is LCFS complete?
  - not in general: a cycle with zero or negative arc costs could be followed forever.
  - yes, as long as arc costs are strictly positive
- What is the time complexity, if the maximum path length is m and the maximum branching factor is b?
  - The time complexity is $O(b^m)$: must examine every node in the tree.
  - Knowing costs doesn't help here.
- What is the space complexity?
  - Space complexity is $O(b^m)$: we must store the whole frontier in memory.
- Is LCFS optimal?
  - Not in general. Why not?

Analysis of Lowest-Cost-First Search

- Is LCFS complete?
  - not in general: a cycle with zero or negative arc costs could be followed forever.
  - yes, as long as arc costs are strictly positive
- What is the time complexity, if the maximum path length is m and the maximum branching factor is b?
  - The time complexity is $O(b^m)$: must examine every node in the tree.
  - Knowing costs doesn't help here.
- What is the space complexity?
  - Space complexity is $O(b^m)$: we must store the whole frontier in memory.
- Is LCFS optimal?
  - Not in general. Why not?
  - Arc costs could be negative: a path that initially looks high-cost could end up getting a "refund".
  - However, LCFS *is* optimal if arc costs are guaranteed to be non-negative.