Uninformed Search

CPSC 322 Lecture 5

September 14, 2007
Textbook §2.4
Lecture Overview

1. Graph Search
2. Searching
3. Depth-First Search
Search

- What we want to be able to do:
  - find a solution when we are not given an algorithm to solve a problem, but only a specification of what a solution looks like
  - idea: search for a solution

Definition (search problem)

A search problem is defined by

- A set of states
- A start state
- A goal state or goal test
  - a boolean function which tells us whether a given state is a goal state
- A successor function
  - a mapping from a state to a set of new states
Abstract Definition

How to search

- Start at the start state
- Consider the different states that could be encountered by moving from a state that has been previously expanded
- Stop when a goal state is encountered

To make this more formal, we’ll need to talk about graphs...
Search Graphs

Definition (graph)
A graph consists of
- a set $N$ of nodes;
- a set $A$ of ordered pairs of nodes, called arcs or edges.

Node $n_2$ is a neighbor of $n_1$ if there is an arc from $n_1$ to $n_2$.
  i.e., if $\langle n_1, n_2 \rangle \in A$

Definition (path)
A path is a sequence of nodes $\langle n_0, n_1, \ldots, n_k \rangle$ such that
$\langle n_{i-1}, n_i \rangle \in A$.

Definition (solution)
Given a start node and a set of goal nodes, a solution is a path from the start node to a goal node.
Example Domain for the Delivery Robot

The agent starts outside room 103, and wants to end up inside room 123.
Graph Search

Searching

Depth-First Search

Example Graph for the Delivery Robot

Uninformed Search

CPSC 322 Lecture 5, Slide 7
Graph Searching

- Generic search algorithm: given a graph, start nodes, and goal nodes, incrementally explore paths from the start nodes.
- Maintain a frontier of paths from the start node that have been explored.
- As search proceeds, the frontier expands into the unexplored nodes until a goal node is encountered.
Problem Solving by Graph Searching

Uninformed Search
Graph Searching

- Generic search algorithm: given a graph, start nodes, and goal nodes, incrementally explore paths from the start nodes.
- Maintain a **frontier** of paths from the start node that have been explored.
- As search proceeds, the frontier expands into the unexplored nodes until a goal node is encountered.
- The way in which the frontier is expanded defines the search strategy.
Lecture Overview

1. Graph Search
2. Searching
3. Depth-First Search
Graph Search Algorithm

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

### Algorithm

- \( \text{frontier} := \{ \langle s \rangle : s \text{ is a start node} \} \);
- 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`;

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.
Branching Factor

Definition (forward branching factor)

The **forward branching factor** of a node is the number of arcs going out of that node.

Definition (backward branching factor)

The **backward branching factor** of a node is the number of arcs going into the node.

If the forward branching factor of every node is $b$ and the graph is a tree, how many nodes are exactly $n$ steps away from the start node?
**Branching Factor**

**Definition (forward branching factor)**

The **forward branching factor** of a node is the number of arcs going out of that node.

**Definition (backward branching factor)**

The **backward branching factor** of a node is the number of arcs going into the node.

- If the forward branching factor of every node is $b$ and the graph is a tree, how many nodes are exactly $n$ steps away from the start node?
  - $b^n$ nodes.

- We’ll assume that all branching factors are finite.
Comparing Algorithms

Definition (complete)
A search algorithm is **complete** if, whenever at least one solution exists, the algorithm is guaranteed to find a solution within a finite amount of time.

Definition (time complexity)
The **time complexity** of a search algorithm is an expression for the worst-case amount of time it will take to run, expressed in terms of the maximum path length $m$ and the maximum branching factor $b$.

Definition (space complexity)
The **space complexity** of a search algorithm is an expression for the worst-case amount of memory that the algorithm will use, expressed in terms of $m$ and $b$. 
Lecture Overview

1. Graph Search
2. Searching
3. Depth-First Search
Depth-first Search

- Depth-first search treats the frontier as a stack
- It always selects one of the last 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 $p_1$ replace $p_1$ at the beginning of the frontier.
- Thus, the frontier is now $[(p_1, n_1), \ldots, (p_1, n_k), p_2, \ldots, p_r]$.
- $p_2$ is only selected when all paths extending $p_1$ have been explored.
Illustrative Graph — Depth-first Search Frontier
Analysis of Depth-first Search

- Is DFS **complete**?

Depth-first search isn’t guaranteed to halt on infinite graphs or on graphs with cycles. However, DFS is complete for finite trees.

What is the time complexity, if the maximum path length is $m$ and the maximum branching factor is $b$? The time complexity is $O(bm)$: must examine every node in the tree.

Search is unconstrained by the goal until it happens to stumble on the goal.

What is the space complexity? Space complexity is $O(bm)$: the longest possible path is $m$, and for every node in that path must maintain a fringe of size $b$. 
Is DFS complete?

Depth-first search isn't guaranteed to halt on infinite graphs or on graphs with cycles.
However, DFS is complete for finite trees.

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

The time complexity is \( O(bm) \): must examine every node in the tree.

Search is unconstrained by the goal until it happens to stumble on the goal.

What is the space complexity?

Space complexity is \( O(bm) \): the longest possible path is \( m \), and for every node in that path must maintain a fringe of size \( b \).
Analysis of Depth-first Search

- Is DFS **complete**?
  - Depth-first search isn't guaranteed to halt on infinite graphs or on graphs with cycles.
  - However, DFS *is* complete for finite trees.
- 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.

Search is unconstrained by the goal until it happens to stumble on the goal.

What is the space complexity?

Space complexity is $O(bm)$: the longest possible path is $m$, and for every node in that path must maintain a fringe of size $b$. 

Analysis of Depth-first Search

- Is DFS **complete**?
  - Depth-first search isn't guaranteed to halt on infinite graphs or on graphs with cycles.
  - However, DFS *is* complete for finite trees.

- 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.
  - Search is unconstrained by the goal until it happens to stumble on the goal.
Analysis of Depth-first Search

- **Is DFS complete?**
  - Depth-first search isn't guaranteed to halt on infinite graphs or on graphs with cycles.
  - However, DFS *is* complete for finite trees.

- **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.
  - Search is unconstrained by the goal until it happens to stumble on the goal.

- **What is the space complexity?**
Analysis of Depth-first Search

- **Is DFS complete?**
  - Depth-first search isn't guaranteed to halt on infinite graphs or on graphs with cycles.
  - However, DFS is complete for finite trees.

- **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.
  - Search is unconstrained by the goal until it happens to stumble on the goal.

- **What is the space complexity?**
  - Space complexity is \( O(bm) \): the longest possible path is \( m \), and for every node in that path must maintain a fringe of size \( b \).