Tree-Based Search

Robot Navigation Problem

Jacob Milligan
100660682

Swinburne University of Technology
COS30019 - Introduction to AI

April, 2017, Semester 1

Acknowledgements

I would like to acknowledge the work of Russell and Norvig (Russell and Norvig 2010), whose book 'Artificial Intelligence: A Modern Approach' was referenced significantly during both research and implementation of the search program. I would also like to acknowledge the work of Amit Patel, found at Red Blob Games, whose writings on A* and heuristics-based search assisted greatly in understanding these algorithms (Patel 2014).

Software Libraries and Code Repository

The code-base is version controlled via git and can be accessed on github and uses several external libraries. These include:

- Catch - A header-only unit-testing framework for C++
- SDL2 - Platform abstraction media library used in the visualizer for opening a window, handling input and rendering graphics
- SDL2_ttf - TrueType rendering library extension for SDL2 for drawing text
- Skyrocket.Path - The platform agnostic file-system component from my game framework, Skyrocket, for navigating the file-system with ease independent of the operating system being used.

Contents

1. Introduction **1**

2. Search Algorithms **1**
2.1. Breadth-First Search (BFS) 1
2.2. Depth-First Search (DFS) 1
2.3. Greedy Best-First Search (GBFS) 2
2.4. A* Search (A*) 2
2.5. Iterative-Deepening Depth-First Search (IDDFS) 2

Acronyms **3**

Glossary **3**

Appendix A. Stuff **4**

1. Introduction

Many generalized search algorithms exist that can be applied to both graph and tree-based problems. These are often broadly divided into two categories - Uninformed search, and Informed search. This report explores six such search methods - Breadth-First Search (BFS) (uninformed), Depth-First Search (DFS) (uninformed), Greedy Best-First Search (GBFS) (informed), A* Search (A*) (informed), Iterative-Deepening Depth-First Search (IDDFS) (uninformed), Iterative-Deepening A* Search (IDA*) (informed) - their completeness, optimality, and efficiency.

Each algorithm is applied to the Robot Navigation Problem (RNP) - a Task environment in which a simulated robot attempts to find an optimal route to a given *goal* node in an $N \times M$ grid of nodes, where $N = width$, $M = height$ and both are non-zero. Furthermore, a GUI-based program for building different states of the RNP, executing different search algorithms on them, and displaying information about execution time, node expanded, and path length has been developed for the purposes of demonstrating visually the content of this report.

Glossary of Terms The concepts and terminology referenced here as they relate to graph and tree based search are defined and described for the reader in the 3 section. Links to these entries are found throughout the report, highlighted blue for reference.

2. Search Algorithms

Search algorithms are generally implemented by building a search tree, each node corresponding to a *state*; a location in the RNP grid. As the algorithm progresses each node adjacent to the current state, starting from the starting location, is placed into a Frontier, which stores all encountered but unexpanded nodes thus far. Once all adjacent nodes have been placed into the frontier, they are taken off it, each of their children expanded and analysed in an order specified by the algorithm. Once the goal state has been found, the parent of each node from the goal will be traversed until no parent is found, creating a direct path from the start location.

The following six search algorithms were implemented to provide solutions to the RNP, three of which are informed search methods and three uninformed search methods.

2.1. Breadth-First Search (BFS)

Uses a First-In, First-Out queue structure (FIFO) as its frontier. This results in each node being expanded in the order it was encountered, meaning the shallowest node in the tree will always be expanded first. In both graph and tree search, BFS is

2.2. Depth-First Search (DFS)

Uses a Last-In, First-Out queue structure (LIFO) as its frontier, which results in the *deepest node encountered thus far* always being expanded first.

2.3. Greedy Best-First Search (GBFS)

A basic informed search method whose evaluation function is denoted as $f(n) = h(n)$ where $h(n)$ is a heuristic function that returns the cost to get from the current node to the goal. Uses a priority queue ordered by lowest f first as its frontier.

2.4. A* Search (A*)

An improved informed search method whose evaluation function $f(n)$ is a combination of both its heuristic and path cost, denoted as $f(n) = g(n) + h(n)$. Uses a priority queue ordered by lowest f first as its frontier.

2.5. Iterative-Deepening Depth-First Search (IDDFS)

An improved uninformed search which runs iterations of DFS, stopping either once the goal is encountered or some depth-limit ℓ is reached, upon which the search tree is cleared, increasing ℓ by 1 and running the algorithm again. This method aims to improve memory

References

Patel, Amit (2014). *Introduction To A**. URL: http://www.redblobgames.com/pathfinding/a-star/introduction.html.

Russell, Stuart and Peter Norvig (2010). *Artificial Intelligence: A Modern Approach*. 3rd Edition. Edinburgh Gate, Harlow, Essex, England: Pearson Education Limited.

Acronyms

A* A* Search.

BFS Breadth-First Search.

DFS Depth-First Search.

FIFO First-In, First-Out queue structure.

GBFS Greedy Best-First Search.

IDA* Iterative-Deepening A* Search.

IDDFS Iterative-Deepening Depth-First Search.

LIFO Last-In, First-Out queue structure.

Glossary

completene A search algorithm is called **complete** when it's guaranteed to find a solution as long as one exists.

evaluation function A function, denoted as $f(n)$ which returns an evaluation of the estimated cost of a specific node n, calculated using either $h(n)$ or some combination of both $h(n)$ **and** $g(n)$.

frontier a container that stores all encountered but unexpanded nodes so far. The data structure used to implement the frontier is specific to each algorithm.

heuristic function A function, denoted as $h(n)$ which returns an *estimation* of the quality of a specific node n.

informed search Synonymous with *heuristic search*. A search algorithm that has domain-specific knowledge of the search space, such as the distance from the start or end states, that define a heuristic function function $f(n)$ for evaluating the quality of a given node n.

node A generalized element of different types data structures that represents a single value or object of interest. It will usually contain a reference or pointer to it's parent or adjacent nodes.

optimal A search algorithm is called **optimal** when it is guaranteed to find the solution with the lowest path cost that exists each time it's executed.

RNP The **Robot Navigation Problem (RNP)** is a Task environment in which a simulated robot attempts to find an optimal route to a given *goal* node in an $N \times M$ grid of nodes, where $N = width$, $M = height$ and both are non-zero.

root node The node at the start of a tree data structure from which all children extend, usually denoted by removing its parent reference, i.e. making it equivalent to null.

task environment In the context of a search problem, the *task environment* is the space and problem for which an agent attempts to find a solution.

tree An abstract data type that represents a hierarchy of nodes extending from a single root node node.

uninformed search A search algorithm that has no information of the search space other than the ability to detect the type of node encountered (i.e. start, end, wall).

Appendix A Stuff

Some appendix stuff should go here