AI: Informed Search to Navigate the Subway

Brian O'Neill

Western New England University
Springfield, MA, USA
brian.oneill@wne.edu

Course Artificial Intelligence

Programming Language Python, Java

Resource Type Assignment

CS Concepts • VI.9.i Graph Operations / Algorithms

Knowledge Unit Programming Concepts

SYNOPSIS

This assignment allows students to gain experience with defining AI search problems and implementing uninformed and informed search algorithms. Students define the search problems for navigating a subway system, requiring them to define the goal test, cost function, and successor function. Students then implement breadth-first search, depth-first search, and A* search. Finally, the assignment requires students to implement a problem in a completely different domain (the 8-puzzle) in order to demonstrate that the search algorithms will work so long as the problem is correctly defined. Students are given data files for the Boston "T" and London "Tube" systems, including functions to parse these data files and build appropriate data structures. This allows students to focus on the search aspects of the problem, rather than implementing the required graph data structures from the raw data.

ACM Reference Format:

O'Neill, B. 2022. AI: Informed Search to Navigate the Subway. In EngageCSEdu. ACM, New York, NY, USA,. 2 pages. https://dx.doi.org/10.1145/3564622

KEYWORDS

Informed search, uninformed search, assignment, artificial intelligence, subway navigation.

1 INTRODUCTION

The purpose of this assignment is to provide students with the opportunity to implement uninformed and informed search algorithms in the context of finding a path between two stations in the Boston and London public transportation systems. For these tasks, the students are provided with functions that construct graphs representing the appropriate transportation networks and methods to query these graphs. Students may therefore focus on defining the search problem and implementing the search algorithms.

Questions 1-5 directly apply to the subway navigation problem. Students begin by defining the search problem itself, constructing the successor function, goal test, and cost function for the general subway navigation task. Students implement algorithms for depth-first search, breadth-first search, and A* search and apply the algorithms to the navigation problem. For A* search, students are told to use straight-line distance as the heuristic, which is easily calculated using code provided with the assignment. Students are then asked to redefine the search problem so that the goal test is satisfied not only by reaching the specified destination, but any station within straight-line distance *d* of that destination. Again, students apply the three search algorithms to the revised problem definition.

In order to demonstrate and reinforce that the search algorithms are independent of the problems they are applied to, Question 6 instructs students to define a new search problem class for the sliding 8-puzzle (again implementing the successors, goal test, and cost functions) and to apply their search algorithms to this problem. Note that the heuristic function required for A* search is defined in each problem class, rather than the search class, further indicating that the search algorithm is independent from the problem.

The assignment concludes with three short-answer questions about uninformed search, informed search, and these particular domains. The most challenging question (item (b)) notes that the subway navigation problem's data sets are flawed, such that the straight-line distance exceeds the actual track distance for some pairs of stations. Students are expected to recognize that the straight-line distance heuristic does not meet the conditions for admissibility, so there is a chance that the results reported by A* search are sub-optimal.

2 ENGAGEMENT HIGHLIGHTS

This assignment uses Meaningful and Relevant Content. Navigating an unfamiliar transportation system, or more broadly, an unfamiliar city, is a real problem. Students have personal experience using navigation software (e.g. Google Maps). While they likely have used it more for driving directions, faculty can point out that Google provides public transit directions for most major cities, Boston and London included. Navigating a known network is a classic AI problem, and applying uninformed and informed search algorithms to the challenge of navigating a subway system is an example that is readily clear and meaningful to students.

This work is licensed under a Creative Commons Attribution 4.0 International License.
ACM EngageCSEdu, September 2022.
©2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9925-8/22/09.
DOI: http://dx.doi.org/10.1145/3564622

To make this assignment more meaningful to their own students, faculty may want to adopt other transportation networks besides Boston and London. Changing at least one of the subway systems to one that is closer to your school would add meaning to the content. For schools located some distance from a subway system, a small bus system (even an on-campus bus system) would be a reasonable substitute for the Boston network.

3 RECOMMENDATIONS

This assignment was developed for an upper-level AI class where search is the first major topic covered. We announce the assignment as we shift from uninformed search to informed search, allowing two to three weeks to complete the assignment.

The assignment contains starter code for both Python and Java. We include both because most of our students have experience in both languages and we want to encourage them to use their preferred language. Adopters could offer support in only one of the two languages without impacting the merits or content of the assignment.

As noted above, faculty adopting this assignment may want to change the subway systems used to be more relevant to their students. The Boston subway system was chosen because it is one that our students are somewhat likely to have used themselves. It is also good for debugging, since the hub-and-spoke nature of the system means that there are few paths between any given pair of stations. In contrast, the London system is more complex and has many paths available. Faculty considering replacing either network should consider the new network's complexity when deciding which network to remove. Adopters should also note that building the data sets for both systems required compiling data from multiple sources; the same would likely be true for any other desired transit network. The readme file in the data folder cites the sources for both datasets.

4 MATERIALS

The Project 1 Word file is the student-facing assignment instructions. It contains a summary, assigned questions, and a brief description of the code files given to students and the functions therein. A grading rubric file is also included for instructors.

The data folder contains five files. The first four are comma-separated variable (CSV) files describing the structure of the Boston and London public transit systems. For each system, one CSV file gives a list of stations and their latitude and longitude coordinates. (Latitude and longitude are used to calculate straight-line distance between the stations as a heuristic for A* search.) The other CSV file for each system is effectively an adjacency list, including track distances between each pair of adjacent stations. The final file is a readme.txt file which explains the construction of the other four files, including sources of data.

The python folder contains two files. The subway.py file defines classes for Links, Stations, and SubwayMaps. These are effectively Edge, Vertex, and Graph classes, with appropriate functions. There are also static functions to construct the Boston and London subway maps (graphs) from the data sets provided. Additionally, we provide a function to calculate straight-line distance between two latitude/longitude points for use with A* search.

The search.py file is an adaptation of code distributed alongside Russell & Norvig's *Artificial Intelligence: A Modern Approach* textbook [1, 2]. (Note that their code was distributed under the MIT license.) The code in this file defines an abstract Problem class and a class for a Node in a search tree. It also provides stub methods for each search algorithm that students are expected to implement.

The java folder provides similar code to the python folder. Because of the nature of the language, each Python class described above is in an individual Java file, belonging to either the search or subway package. The Java code for the classes in the search package are based on Russell & Norvig's Python code [2], rather than their Java code, so that the code distributed with this assignment would have parallel structures across languages. Additional classes are provided to represent Actions and States (as related to search problems) and Tuples (a structure to unite Actions and States). These classes are necessary in Java due to the type system; various functions in the Problem and Node classes require Actions or States. In Python, students can use strings, simple data structures, or custom objects to represent states and actions.

5 PITFALLS

Despite guidance in Question 1 that directs students to the subway package (or the equivalent Python script), including the existence of functions that will create the graph for the two transportation networks, some students end up writing their own code to read the subway data files to build the networks instead of using those functions. This is a major time sink for the students, particularly if they end up building the network incorrectly. We have considered reminding students during class that these functions exist and all of their efforts should be on designing the problems and implementing the algorithms.

One issue that can arise during grading, or earlier if students are comparing notes on their results, is that different orderings within the student-built `successors()` function in Question 1 can yield substantially different results for depth-first search and breadth-first search in Questions 2-3. The different orderings result from the use of different provided graph methods to identify neighboring stations. On the student side, this can lead them to believe that their algorithm or problem definition are incorrect when they are not. On the grading side, faculty need to be attentive to the order that neighboring states were generated in the `successors()` function while checking the accuracy of DFS and BFS solutions. Question 6 avoids this issue by specifying an ordering for neighbors; this is only reasonable because the smaller domain makes a consistent ordering simpler to accomplish. Establishing a consistent ordering for the subway networks would be a much greater challenge.

6 REFERENCES

[1] Stuart J. Russell and Peter Norvig. 2009. *Artificial Intelligence: A Modern Approach* (3rd ed.). Prentice Hall.

[2] Peter Norvig. 2021. Github: aimacode. Code for the book "Artificial Intelligence: A Modern Approach." Retrieved from https://github.com/aimacode/.