AI Search Algorithms

Capstone Final Report

Author: Elicia Pluymers

Faculty Advisor: Dr. Mohammad Al Hasan

Faculty Advisor Email: [alhasan@iupui.edu](mailto:alhasan@iupui.edu)

Abstract

The following document reviews the details of the AI Search Algorithms Program implementation and design. The program goal design was to implement a program, coded using Java, that will implement a variety of algorithms using more than one form of data. The final program was meant to be modular and potentially expandable in the future, to allow for additional algorithms and data forms to be implemented using the base program. The final program is functional and modular, though there is room for improvement. The program successfully displays a graphical representation of two forms of data, a linked node graph displayed in tree search form and a n x n board chess board with queens placed on the board. The program also successfully implements a variety of search algorithms for each graphical representation. Data is loaded in the program using a user defined text document which allows for users to enter their own data to test and view how each algorithm iterates through its solution algorithm.

Overall, the program performs as expected, but the program still experiences difficulties following some of the linked node graphs that have been tested. Further details regarding these issues are described in detail below. Some algorithms have not been implemented but the structure is in the database for future expansion and further work. In regards to modularity the program does implement the data separately from the algorithms and utilizes abstracted classes and interfaces to communicate between the two.

Introduction

The main objective of this project was to design a functional and well-designed program that has the potential to be a teaching tool for introductory Artificial Intelligence courses. Due to the limited time available within a single semester long course, many courses expect implementation of only one or two of the many search algorithms that are discussed in the course. Through my experience this is the case for many courses all due to the same issue, limited time. It would be advantageous for many students to be able to have hands-on experience with these algorithms to assist in a deeper understanding of how they function.

The goal is not to implement the algorithms in code for a specific dataset or goal, but to create an abstracted framework that will enable the reuse of a base program in the implementation process while still allowing for flexibility in state-space and algorithm choice. If time allows the plan is to also implement a GUI based display vs a command line system, to assist in visualization of how a search algorithm is navigating the state-space. This project will require a detailed plan to implement in a way that is functional and easy to use. I will be utilizing software design and project management skills to create the design and implement the coding aspect while also developing a deeper understanding of the search algorithms that will be implemented as part of this process.

Problem

There is a large amount of information on implementing specific search algorithms through object-oriented and also functional programming languages. In an article on Functional implementations of these algorithms the concept of arranging a hierarchy of related algorithms was discussed and also implemented previously in Java [1]. Additionally, there are many textbooks and language-based texts that instruct how to implement these algorithms, but does not discuss implementing the structure to allow for reuse of code or modular design [2-4].

In order to approach this problem, I will be reviewing prior work on object-oriented implementations of these algorithms and determining the best way to apply software design patterns to implement a program that will be more modular in design.

The goal of this project is to produce a functioning program, that can successfully implement the basic artificial intelligence search algorithms and display those results through a graphical interface. The algorithms that were part of the initial design goals are listed below.

Uninformed Search Algorithms

* Breadth-First
* Uniform-Cost
* Depth-First
* Depth-Limited
* Iterative Deepening Search
* Bidirectional Search

Informed Search Algorithms

* Greedy Best First
* A\* Search

Local Search Algorithms

* Hill-Climbing Search
* Simulated Annealing
* Local Beam Search

This project will require implementation of software design patterns, understanding of AI search algorithms, full stack program design, and project planning and management. In completing this project I have strived to develop a deeper understanding of software development as a whole and an understanding of the practical use of many of the key ideas and concepts learned during my undergraduate studies.

Design

Within the design of this program, I chose to utilize program design techniques to design and write the code. Utilized within this code is the Builder Pattern, Abstract Factory Pattern, and Observer Pattern. I also utilized abstract classes and interfaces where possible to make the accessing of data from other classes uniform and not require the calling class to know details of the specific class it was calling.

The final project is split into three separate packages, each shown below as a class diagram, containing closely correlated data. With one exception, communication between these packages is mostly limited to predefined interface function calls or abstract functions. The one exception is communication between the algorithm class and it’s related data class, as I was unable to find a functional way to fully split the algorithm’s function calls to the data class without either limiting the number of algorithms that the program could support, or supplying information about the type of algorithm being used to the data class which as a part of the project was trying to be avoided. For future work, I wish to work on a better model to further abstract out the data from the interface used to access information needed to complete the algorithm portion of the system.

A diagram of a program

Description automatically generated with low confidence

Within the userInterface package all user input data is retrieved from the user and sent to the necessary portions of the other packages. As part of good design practices, the user only ever interacts with the userInterface package classes.

A picture containing text, diagram, plan, technical drawing

Description automatically generated

Within the dataStorage package all classes related to storage, processing and visualization of the data is kept. This package has a builder class which calls an abstract factory to parse the user input sent from the userInterface. This factory creates the data and also links an imageInterface to the data it will be displaying the image for before passing both back to the main builder class. This package also utilizes the observer pattern to allow the image handler to update the data visualization after each step or run from the user interface.

A picture containing text, screenshot, number, parallel

Description automatically generated

Within the algorithmData package all classes are related to the algorithms that will act on the user entered data. This package contains a builder which initializes the user selected algorithm and links it to the data that it will be working on. Through the implementation process I was most dissatisfied with the way this was constructed as the goal would have been to better abtract the communication between the data and the algorithm.

Implementation

After a rough design plan was created, implementation began. Below is an image of the Gantt chart created at the beginning of the project which tracks when I began and completed each of the preset segments of the project.

A screenshot of a project planner

Description automatically generated with medium confidence

In review of the design and implementation plan determined at the beginning of the project, I began some steps much sooner than planned as I adjusted for the overall scope of the project. After the preliminary design, I began working on the user interface almost immediately, vs the expectation that I would be completing that portion of the project closer to the end of the implementation phase. Additionally, implementation of the algorithms and data storage began later than expected. This was tied to the fact that after design I was concerned with how the GUI for the user data would be displayed and wanted to ensure that I did not need to redesign the entire data visualization class after the user interface was completed.

Error Handling

In order to ensure the program, is it does encounter and error, can fail gracefully some custom exception classes were created. These exceptions are thrown by classes handling user input, which due to the fact that users can generate and define their own text-based data file, cannot be fully eliminated. Instead of limiting the ability of users to generate their own data, the exceptions were put in place to allow the program to fail, but give the user output informing them of the issue. These exceptions are caught and an error message is sent to be displayed to the end user after the program closes. Additionally, for the final program and unforeseen errors that were not seen during testing are caught by a blanket exception handler in the main program. This is primarily to avoid the end user from needing to see error stack traces.

Program Execution and Limitations

The program successfully executes the following algorithms.

Graph Data:

Uninformed Search Algorithms

* Breadth-First
* Uniform-Cost
* Depth-First
* Depth-Limited
* Iterative Deepening Search
* Depth-Limited
* Iterative Deepening Search

Informed Search Algorithms

* Greedy Best First
* A\* Search

NQueens Data:

Uninformed Search Algorithms

* Breadth-First
* Depth-First

Informed Search Algorithms

* Greedy Best First

In designing the program all algorithms were set up within the program, these algorithms are set to throw an error that is handled by the program if they are selected before having been implemented and the line that throws the exception being removed. The program operates even with these algorithms not being implemented. In future updates there is an opportunity to implement these algorithms without any major changes to the remainder of the code.

The graph-based visualization was set with certain tolerances to fit within a standard view window. It is best at displaying nodes with up to 6 forward connections, after which there is some overlap between the displayed nodes depending on the node name length. The visualization also displays the cost associated with each node, the cumulative cost for each node in the solution set and the heuristic value of each node. Some of this data is not used for all algorithms but is displayed for every graph representation. Because of the limit to the number of nodes that can be displayed in the image window, only nodes along the solution path are expanded and visible.

The program window displays when the algorithm fails to find a solution as well as when it succeeds and stops the user from selecting the run and step buttons as the algorithm has completed. This is to reduce errors as well as allow the user to run the program for a number of iterations automatically until a solution is found of the number of iterations has occurred. After the algorithm completes, success or failure the amount of iterations required to reach that state are displayed in the window.

Screen Shot Appendix

File Loading Screen

A screenshot of a computer

Description automatically generated with medium confidence

User Algorithm GUI for Graph

A screenshot of a computer

Description automatically generated

User Algorithm GUI for Graph

Breath First Search After Algorithm Completion

A screenshot of a computer

Description automatically generated

User Algorithm GUI for NQueens

A screenshot of a computer

Description automatically generated

User Algorithm GUI for NQueens

Breath First Search After Algorithm Completion

A screenshot of a game

Description automatically generated

References

[1] J. Pánovics, “A Functional Programming Approach to AI Search Algorithms,” *Journal of Information*

*Technology Education: Innovations in Practice*, vol. 11, pp. 353–376, 2012, doi: 10.28945/1736.

[2] G. F. Luger and W. A. Stubblefield, *AI algorithms, data structures, and idioms in Prolog, Lisp, and Java*.

Boston, Ma: Pearson Addison-Wesley, 2009.

[3] A. Artasanchez and P. Joshi, *Artificial intelligence with Python : your complete guide to building*

*intelligent apps using Python 3.x and TensorFlow 2*. Birmingham: Packt Publishing, 2020.

[4] S. Russel and P. Norvig, *Artificial Intelligence : A Modern approach.*, 4th ed. Prentice Hall, 2018.