**Puzzle Sliding Game**

COMP 8700 Fall 2020 Final Project

**Xinmeng Yuan,**

Master of Applied Computing,

University of Windsor,

yuan12b@uwindsor.ca

**Ruiting Liu,**

Master of Applied Computing,

University of Windsor,

liu72@uwindsor.ca

**Xinyu  Ji**,

Master of Applied Computing,

University of Windsor,

ji11p@uwindsor.ca

**Mengyao Liao,**

Master of Applied Computing,

University of Windsor,

liao71@uwindsor.ca

**Wen Dong**,

Master of Applied Computing,

University of Windsor,

dong23@uwindsor.ca

***Abstract –*** this is the report of our team final project of Artificial Intelligence Introduction, in this project, we have experimented and explored a set of search algorithms including informed and uninformed ones with the famous Puzzle Sliding Game, and found that the performance of different algorithms varies markedly in terms of optimality, completeness, time complexity and space complexity; We also found that heuristic function plays a significant role in the search for informed algorithms, and impact the outcome tremendously.

**Introduction**

The project is the final project of our AI Introduction course, which is to put the comprehensive knowledge and theory we have learned from class into reality, from abstract intuition to tangible experiment. It is beneficial for all the team members to consolidate the learning and build hands-on experience on Artificial Intelligence.

In the project, we have chosen and explored a set of informed algorithms:

* AStarSearch – A graph traversal and path search algorithm, an import and widely used search algorithm in computer science due to its completeness, optimality, and optimal efficiency. It uses function f(n) = g(n) + h(n) as estimate function, based on which to expand cheapest node first. The sole drawback is its space complexity therefore not practical for large-scale problems.
* RecursiveBestFirstSearch – It is a variant version of A Star for large-scale problems where A Star is unable to tackle due to space complexity. Instead of keeping all the frontiers in memory, it uses a flag f\_limit to keep track of the f-value of the best alternative path available from any ancestor of the current node.
* GreedyBestFirstSearch – It is designed to find goal quickly by expanding the node that is closest to the goal, on the grounds that this is likely to lead to a solution quickly with evaluate function f(n) = h(n). So, nodes not on the solution path are not expanded. Cost low, however, it is not optimal neither complete because optimal solution might be in other unexplored paths.

Additionally, two uninformed as well:

* BreadthFirstSearch – Breadth-First is a search strategy in which the nodes are expanded and explored in a breadth first way, that is the root node expanded, then the successors are expanded and so on. When all steps costs are equal, it is optimal as it always expands the shallowest unexpanded node.
* UniformCostSearch – with a simple extension on Breadth-First search, Uniform-Cost Search is optimal for any step-cost function. Instead of expanding the shallowest node, it expands the node with lowest path cost g(n)

Aside from the variety of algorithms, we have also employed three different heuristic functions and two different accessory strategies, i.e. tree search and graph search to work with the informed searches, to observe how and how much they impact the searching process.

**Relevant Literature Review**

Studying Puzzle Sliding Game is an effective approach to explore various algorithm, particularly in searching algorithm. The paper Using Puzzles in Teaching Algorithms by Anany, and Mary-Angela (2002) conclude that the teaching design and analysis of algorithm could be implemented by using puzzles. Edmund A. Lamagna (2017) claimed that puzzles and frogs question provide a rich source for algorithmic thinking and improve analytical thinking. There are basic logic thinking problems in search algorithms. The puzzle game promotes the use of algorithm analysis. Most cases could be solved by observing patterns. For example, Sudoku is a popular puzzle game with mixed numbers, which is widely published around the world in online blogs and newspapers. The paper Genetic Algorithms with Local Optima Handling to Solve Sudoku Puzzles by Firas, Germain, and Fanielle (2018) introduced that an NP-Complete problem should be solving in nature heuristic. A genetic algorithm with modified crossover and mutation operators is the heuristic. To solve the different puzzle problems of different levels of difficulty, the designed various approaches algorithms should be tested in existing methods. In addition, the problem of N- puzzles is an important problem in mathematics and has artificial intelligence implications, especially in gaming. Bhasin and Singla (2012) presented how genetic algorithm is applied and studied the influence of solving NP problems by using heuristic algorithms in the paper Genetic based Algorithm for N-Puzzle Problem. To solve combinatorial optimization issues, the A\* algorithm is commonly used, but it requires high computational power and a large amount of memory. In this sense, in order to benefit from the processing power and the accumulated memory given by clusters, Hash Distributed A\* (HDA\*) parallelizes A\*. Scalability analysis of Hash Distributed A\* on commodity cluster: results on the 15-puzzle problem by Sanz, De Giusti, Armando, and Naiouf (2016) concluded that the experiments carried out on a standard cluster with an Ethernet connection did not take account of the latter application for applications with different heuristic computation times, such as domain-independent planning and the Sliding Puzzle. The Sliding Puzzle has recently gained significance because it applies to specific problems such as shifting pallets in high-density storage warehouses with an automated driven vehicle.

**Experiment**

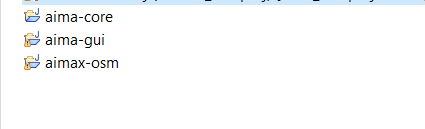
Our experiment is based on the open source project AIMA which is the java implementation of algorithms from Russell and Norvig’s Artificial Intelligence – A Modern Approach 3rd Edition.

Setup

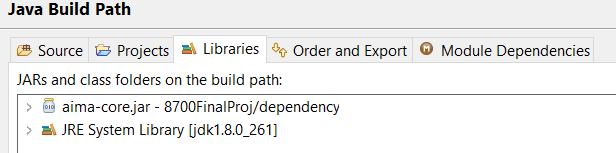
1. Download the latest source code of AIMA java from the GitHub repository

<https://github.com/aimacode/aima-java>

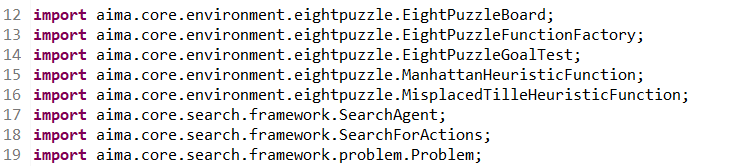
1. open the projects with Eclipse (Version: 2020-09 (4.17.0)) with builder Java 8+



1. Export aima-core as a jar library then add and reference it in our project



1. Finally, we are ready to import all sorts of algorithms and eight puzzles related assets from the library to build our testing flows of interest.



**Coding**

Based on the AIMA project, coding the experimental flows is like a breeze, as most of the work can be achieved simply by permutation and combination of various existential Java classes. For instances, testing A Star with graph search and Manhattan heuristic function is implemented as below:

1. Prepare initial board states to search solution for
2. Create an instance of Manhattan heuristic function class

ManhattanHeuristicFunction hf = new ManhattanHeuristicFunction();

1. Create an instance of A Star search algorithm with the heuristic function and a graph search
2. Assembly the objects together to create a Problem

Problem problem = **new** Problem(initialState,

EightPuzzleFunctionFactory.*getActionsFunction*(),

EightPuzzleFunctionFactory.*getResultFunction*(),

**new** EightPuzzleGoalTest());

1. Instantiate a Search Agent to conduct the actual search and output the search result in the end.

SearchAgent agent = new SearchAgent(problem, search);

printInstrumentation(agent.getInstrumentation());

Find our project repository below, download and run it if you are interested.

<https://github.com/fanchuanster/8700_finalproj.git>

**Testing Results and Analysis**

1. Comparison between algorithms

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Algorithm | pathCost | nodesExpanded | queueSize | maxQueueSize |  |
| Informed | A Star | 18 | 227 | 156 | 157 | Manhattan heuristic function and graph search used. With initial state  5 4 0  3 2 8  6 1 7 |
| Recursive Best First | 18 | 472607 | N/A | 18 |
| Greedy Best First | 58 | 309 | 228 | 229 |
| Uninformed | Breadth First | 18 | 13210 | 7396 | 7396 |  |
| Uniform Cost | 18 | 19268 | 11020 | 11023 |  |

Result Analysis:

From the result, we can see except Greedy Best Frist search, all the others are optimal, among which A Star performs well in the number of expanded nodes, and also with a decent queue size and maximum queue size comparing to uninformed approaches. Uninformed ones are also able to find the optimal solution, but with considerable cost as revealed by nodes expanded and queue size.

1. Comparison between Tree search and Graph search

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Algorithm | | pathCost | nodesExpanded | queueSize | maxQueueSize |
| A Star | Graph Search | 18 | 227 | 156 | 157 |
| A Star | Tree Search | 18 | 14653 | 26347 | 26348 |
| Greedy Best First | Tree Search | Endless searching | | | |
| Greedy Best First | Graph Search | 58 | 309 | 228 | 229 |

Result Analysis:

From the result, we can see Graph search helps reducing the number of expanded nodes noticeably comparing to Tree Search for A Star, and for Greedy Best First search, Graph Search ended with modest cost, while Tree search ran into an endless probe.

1. Comparison between heuristic functions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Heuristic Function | pathCost | nodesExpanded | queueSize | maxQueueSize |
| Manhattan | 18 | 227 | 156 | 157 |
| MisplacedTille | 18 | 1239 | 757 | 758 |
| EuclideanDistance | 18 | 339 | 230 | 231 |

Result Analysis:

For this experiment, three heuristic functions have been tested as we can see above, two of them were from AIMA library and one was created one our own. although all of them are admissive and consistent and able to identify the optimal solution, the matrix of each search varies. It shows that heuristic function is extremely important for an informed search, and it is crucial to pick an appropriate one which can reflect the real situation between the source state and the destinate goal.

Interesting found in AIMA

An AIMA Bug?

During our work, we found many states which turns out to be half of the possible states do not have a solution path towards the goal state, 181440 states. For example:

187432650, 187432605, 187402635

107482635, 017482635, 187042635, …

And the other half 181440 are resolvable, for example:

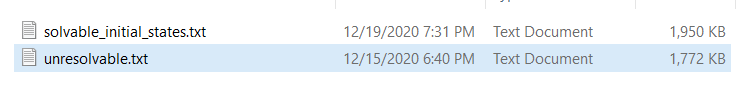
436785102, 236154780, 327186450

605237184, 673805142, 140562378, …

The 181440 \* 2 happens to be the factorial of the length of the game board size 9, this reminds us there should be something wrong with the Eight Puzzles Game in the AIM library.

What have we done to overcome the bug?

To proceed with our work smoothly, we distinguished all the unresolvable states from the solvables to ensure valid ones are selected and tested.



Meanwhile, we reported the issue to AIMA Team in hope a fix would be implemented in the future.

<https://github.com/aimacode/aima-java/issues/478>

**Conclusion and Future Work**

By exploring different approaches with Eight Puzzle game, we have found that different algorithms perform variously due to the nature of each, in general, informed algorithms excel in performance given a suitable heuristic function. Verified that Tree Search and Graph Search influence the performance matrix in terms of node expanded and queue size during the search. Tested and appreciated that heuristic function plays a significant role in informed searches, and it is crucial to identify a well-suited heuristic function for a well-behaved path finding solution.

In the future, we consider intensifying our work deep into more test states, as well as extending our effort to more search algorithms learned from the class, like Depth First Search, Bidirectional Search, Hill Climbing family, and Genetic Search, ex cetera. We believe that will benefit our learning in the field and consolidate our knowledge gained from the study journey.

**Work distribution among team members**

Yuanxin Men – beginning and abstract

Wen Dong – setup and coding

Ruiting Liu, Xinyu Ji – introduction and algorithms selection

Mengyao Liao – testing result analysis and conclusion

**References:**

1. Aimacode. (n.d.). Aimacode/aima-java. Retrieved January 05, 2021, from <https://github.com/aimacode/aima-java>
2. Anany Levitin and Mary-Angela Papalaskari. 2002. Using puzzles in teaching algorithms. SIGCSE Bull. 34, 1 (March 2002), 292–296. DOI: <https://doi.org/10.1145/563517.563456>
3. A freshman seminar on problem solving and algorithmic thinking. (2014). Journal of Computing Sciences in Colleges., 29(6), 29–38. <https://doiorg.ledproxy2.uwindsor.ca/10.5555/2602724.2602732>
4. A\* search algorithm. (2021, January 05). Retrieved January 05, 2021, from [https://en.wikipedia.org/wiki/A\*\_search\_algorithm](https://en.wikipedia.org/wiki/A*_search_algorithm)
5. Bhasin, H., & Singla, N. (2012). Genetic based algorithm for N - puzzle problem. International Journal of Computer Applications, 51(22) doi: <http://dx.doi.org.ledproxy2.uwindsor.ca/10.5120/8347-1894>
6. Edmund A. Lamagna. 2017. Frogs + puzzles = algorithmic thinking. J. Comput. Sci. Coll. 32, 6 (June 2017), 111–119.
7. Firas Gerges, Germain Zouein, and Danielle Azar. 2018. Genetic Algorithms with Local Optima Handling to Solve Sudoku Puzzles. In Proceedings of the 2018 International Conference on Computing and Artificial Intelligence (ICCAI 2018). Association for Computing Machinery, New York, NY, USA, 19–22. DOI: <https://doi.org/10.1145/3194452.3194463>
8. Russell, S. J., &amp; Norvig, P. (2016). Artificial intelligence: A modern approach. Upper Saddle River: Pearson.
9. Sanz, V., De Giusti, A., & Naiouf, M. (2016). Scalability analysis of hash distributed A\* on commodity cluster: Results on the 15-puzzle problem. Athens: The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp). Retrieved from https://search-proquest-com.ledproxy2.uwindsor.ca/conference-papers-proceedings/scalability-analysis-hash-distributed-on/docview/1807220220/se-2?accountid=14789