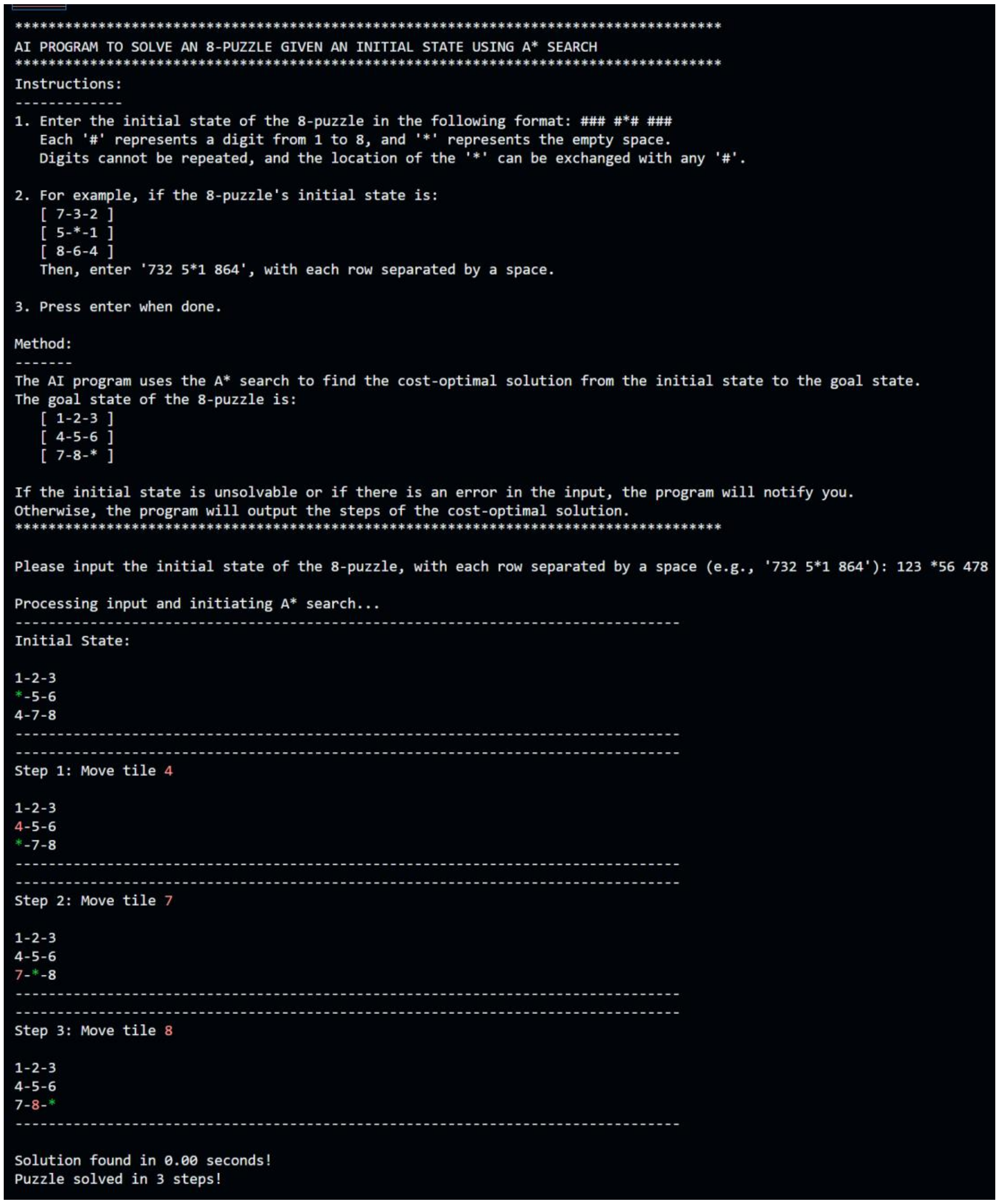
Option #1: Introduction to Informed Search Heuristics: An 8-Puzzle Approach

Scott Miner

Colorado State University – Global Campus

Abstract

This article delves into the application of an informed search algorithm, namely the A* search, to solve the well-known 8-puzzle problem, where the goal is to arrange eight numbered tiles sequentially in a 3x3 grid. The expansive state-space of the 8-puzzle problem necessitates the use of heuristics, which allow for more efficient paths to the solution. The article demonstrates the application of the A* algorithm using a practical example, highlighting its strengths and potential drawbacks. The study also discusses the concept of Multi-Agent Path Finding (MAPF) and draws a parallel with the 8-puzzle problem, suggesting potential real-world applications in areas such as traffic management, warehouse management, and video game design, where strategic planning and efficient use of resources are key.



```
****************************************************************************************
AI PROGRAM TO SOLVE AN 8-PUZZLE GIVEN AN INITIAL STATE USING A* SEARCH
****************************************************************************************
Instructions:
-------------
1. Enter the initial state of the 8-puzzle in the following format: ### #*# ###
   Each '#' represents a digit from 1 to 8, and '*' represents the empty space.
   Digits cannot be repeated, and the location of the '*' can be exchanged with any '#'.

2. For example, if the 8-puzzle's initial state is:
   [ 7-3-2 ]
   [ 5-*-1 ]
   [ 8-6-4 ]
   Then, enter '732 5*1 864', with each row separated by a space.

3. Press enter when done.

Method:
-------
The AI program uses the A* search to find the cost-optimal solution from the initial state to the goal state.
The goal state of the 8-puzzle is:
   [ 1-2-3 ]
   [ 4-5-6 ]
   [ 7-8-* ]

If the initial state is unsolvable or if there is an error in the input, the program will notify you.
Otherwise, the program will output the steps of the cost-optimal solution.
****************************************************************************************

Please input the initial state of the 8-puzzle, with each row separated by a space (e.g., '732 5*1 864'): 123 *56 478

Processing input and initiating A* search...
--------------------------------------------------------------------------------
Initial State:

1-2-3
*-5-6
4-7-8
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Step 1: Move tile 4

1-2-3
4-5-6
*-7-8
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Step 2: Move tile 7

1-2-3
4-5-6
7-*-8
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Step 3: Move tile 8

1-2-3
4-5-6
7-8-*
--------------------------------------------------------------------------------

Solution found in 0.00 seconds!
Puzzle solved in 3 steps!
```

Figure 1. AI program instructions, input, and output to solve an 8-puzzle

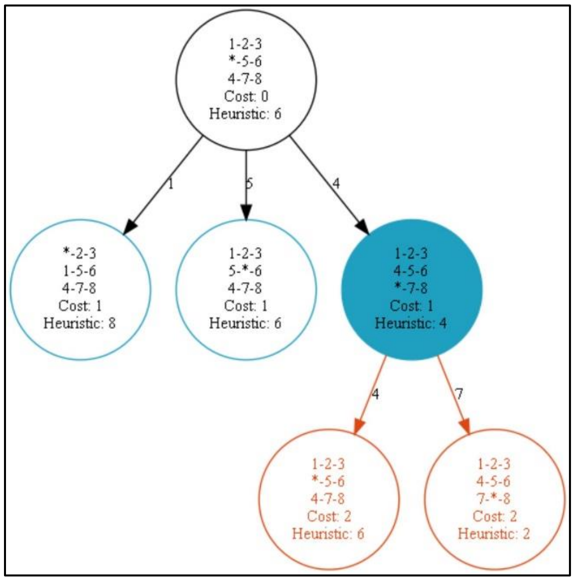


Figure 2. Expanding a search tree using the A* search to solve an 8-puzzle

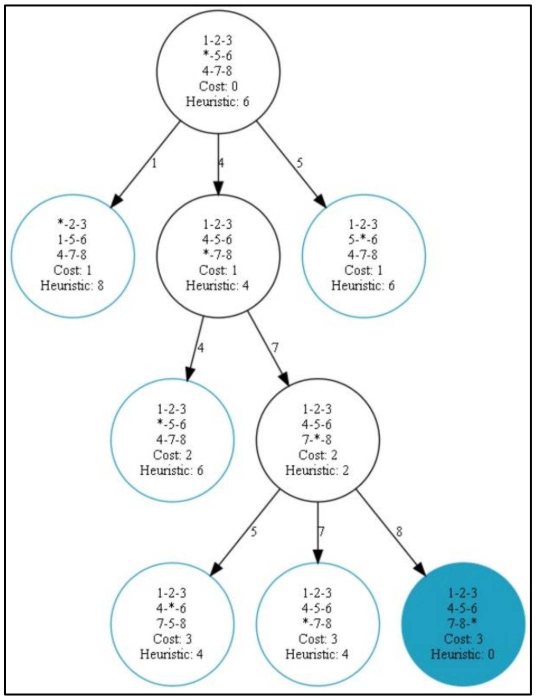


Figure 3. Search tree used by AI program to solve an 8-puzzle

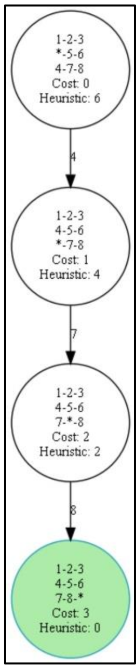


Figure 4. A cost-optimal solution to an 8-puzzle

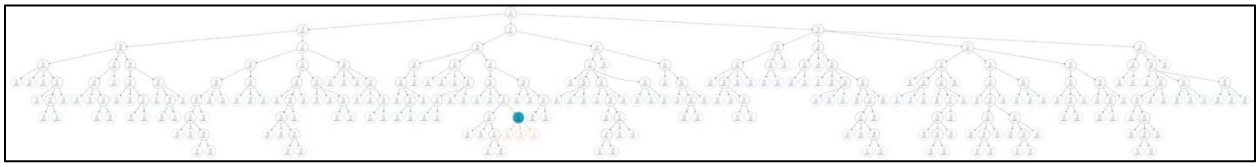


Figure 5. A* search expanding multiple nodes to find a solution

Table of Contents

Option #1: Introduction to Informed Search Heuristics: An 8-Puzzle Approach

The concept of *heuristics*, as described by Feigenbaum and Feldman (1963, as cited in Romanycia & Pelletier, 1985), refers to a procedural method that can potentially resolve a problem, but without providing a guaranteed solution (p. 48). In a more contemporary context, Sharda *et al.* (2020) interpret heuristics as the embodiment of informal knowledge that forms the basis for sound judgment within a particular domain.

This paper sets out to explore a heuristic search problem, modeled on the classic 8-puzzle, a challenging task involving a 3x3 grid filled with eight numbered tiles (1 through 8) and one empty slot. The puzzle demands strategic movements - left, right, up, or down - of the tiles into the blank slot, with the aim of aligning them in an increasing numerical sequence, progressing from left to right and top to bottom.

As noted by Russell and Norvig (2002), each tile movement in the 8-puzzle comes with a cost of one, and the puzzle holds the potential for $9!/2 = 181{,}400$ achievable states. Consequently, the complexity and expansiveness of the 8-puzzle's state-space necessitates the implementation of heuristic search techniques over exhaustive methods (Kunkle, 2001), a concept which will form the crux of this paper's exploration.

Understanding Search Algorithms and State-Space Graphs

According to Russell and Norvig (2002), search algorithms function by superimposing search trees on *state-space graphs* to identify routes from initial states to end goals. In the context of the 8-puzzle, the state-space embodies all potential board configurations, and the search tree provides a roadmap of the transitions between these states.

The Role of Heuristics in Informed Search Algorithms

Russell and Norvig (2002) write that *informed search algorithms* distinguish themselves from *uninformed strategies* through their usage of heuristics. These *heuristics*, or domain-specific hints about goals, enable the algorithms to identify solutions more efficiently by offering insights about the proximity of a given state to the goal.

A* Search: An Optimal Informed Search Algorithm

The most renowned informed search algorithm, as pointed out by Russell and Norvig (2002), is the *A* search* (pronounced "A-star search"). This algorithm, known as a *best-first search*, prioritizes the expansion of nodes that yield the minimum values for its evaluation function, expressed as $f(n) = g(n) + h(n)$. In this function, $g(n)$ represents the cost of the path from the initial state to a particular node, n, while $h(n)$ is the estimated cost of the shortest path from that node to the goal, thus acting as the heuristic. Consequently, $f(n)$ is an estimation of the cost of the most efficient path from a state to the goal.

Ensuring Completeness in A Search*

Russell and Norvig (2002) also highlight the *completeness* of the A* search, meaning that the algorithm is designed to invariably identify a solution if one exists and to accurately declare failure when a solution is absent.

Achieving Optimality in Search Algorithms through Admissible Heuristics

Russell and Norvig (2002) characterize optimal solutions as those having the minimum *path-cost* among all possible solutions. The cost-optimality of the A* search is contingent upon its *heuristic* and its *admissibility*. An *admissible* heuristic is one that never overestimates the cost to reach a goal, whereas *inadmissible* heuristics may exceed the actual cost of reaching the goal, potentially leading to suboptimal solutions. Consequently, when the heuristic applied in an A* search is admissible, the search can be considered cost optimal.

Admissible Heuristics: Misplaced Tiles and Manhattan Distance

Within *n*-puzzle problems, two heuristics are commonly employed: (a) h_1, which represents the number of *misplaced tiles*, excluding the blank tile, and (b) h_2, which is the sum of the distances of the tiles from their goal positions, also known as the *Manhattan distance*. Both h_1 and h_2 are admissible, implying that they do not overestimate the actual cost of a solution.

Comparing Efficiency: A* Search with Misplaced Tiles vs. Manhattan Distance Heuristics

In a comparative analysis, A* search using h_2 consistently outperforms A* search using h_1. This is because h_2 always results in fewer node expansions than h_1. Therefore, the artificial intelligence program outlined in this paper leverages the A* search with the more efficient h_2 heuristic, the Manhattan distance.

An Overview of the A* Search Process in the State-Space Graph

Figure 2 offers a visual representation of the A* search traversing a state-space graph. Nodes characterized by black borders and white backgrounds, such as the root node in this case, denote nodes that are retained in memory. Contrastingly, nodes delineated by blue borders and white backgrounds are yet to be visited.

Understanding Node Expansion and Creation in A Search*

Nodes exhibiting blue borders and blue backgrounds are in the process of expansion, while those with orange borders and white backgrounds represent newly formed nodes following the expansion of a parent node. Each node in the search tree encapsulates the layout of the 8-puzzle, the cost of the node from the initial state, and the node's heuristic. The numbers appearing on the edges leading to the tree vertices identify the tile that was moved to attain each subsequent state.

Visualizing the Goal State in A Search*

Figure 3 exhibits the search tree upon reaching the goal. The node marked by a blue border and blue background in Figure 3 possesses a heuristic value of zero, signifying that this node represents the goal state.

Output Analysis: The Steps to Reach the Goal State

Finally, Figure 4 displays the steps taken to reach this goal. These steps are output to the console by the program as depicted in Figure 1. Furthermore, Figure 1 illustrates the instructions provided by the program.

Overcoming Challenges: Memory Usage and Node Expansion in A* Search

Russell and Norvig (2002) highlight the primary disadvantage of A* search: its considerable memory usage and the extensive number of nodes it expands. For a multitude of problems, the quantity of nodes the algorithm expands can exhibit exponential growth. For instance, Figure 5 illustrates the algorithm visiting over 100 nodes as it seeks a solution to the 8-puzzle.

Recognizing the Strengths of A* Search: Completeness, Cost-Optimality, and Efficiency

The merits of A* search are its completeness, cost-optimality, and optimal efficiency. Russell and Norvig (2002) point out that the algorithm's efficiency stems from its ability to prune search tree nodes unnecessary for identifying an optimal solution. This is a crucial aspect for many domains within AI.

Real-World Applications: Implementing A* Search Beyond the 8-Puzzle

In a purely imaginative real-world scenario, implementing AI solutions to solve an 8-puzzle could prove useful in the field of pathfinding, which involves plotting the most optimal route between two points. This is like the concept of Multi-Agent Path Finding (MAPF) discussed by Felner *et al.* (2017). MAPF is a well-studied problem in AI where the task is to find paths for multiple agents from their start locations to their goal locations without collisions (Felner *et al.*, 2017).

In the context of the 8-puzzle, each agent could represent a tile that needs to move to its goal location (the correct position in the puzzle). The challenge is to find the optimal sequence of moves that will result in all tiles (agents) reaching their goal locations without 'collisions' (i.e., without two tiles trying to occupy the same space at the same time).

Just like MAPF, the 8-puzzle problem requires strategic planning and efficient use of resources. In real-world applications, these AI solutions could be used in various domains such as traffic management, warehouse management, and even in video games, where multiple entities need to navigate a shared space efficiently and without conflict (Felner *et al.*, 2017).

Solving the 8-Puzzle: Considering Inversions and Polarity

To solve an 8-puzzle effectively, two additional terms need to be understood: *inversion* and *polarity*. As defined by Collier (2019), an inversion in an 8-puzzle context is a pair of tiles arranged in descending order instead of the correct ascending order. For example, the pairs (8, 6) and (3, 1) are inversions as they do not follow the correct numerical sequence.

Determining the Solvability of the Puzzle

Collier (2019) writes that the puzzle's *polarity* is determined by the total number of these inversions. Puzzles with an even number of inversions are solvable, whereas those with an odd number of inversions are not. This property of the 8-puzzle is a crucial factor in determining whether a given initial state can be converted to the puzzle's goal state.

Interacting with the AI Program: Input Format and Error Handling

To interact with the AI program, users must input the puzzle's initial state in a specific format: ### #*# ###. Here, tile rows are separated by spaces, and the puzzle's empty tile is represented by an asterisk. The program is designed to handle errors in the input format or the solvability of the initial state. If users enter an initial state that is either in an invalid format or unsolvable, the program alerts the users of their error and prompts them to re-enter the initial state of the 8-puzzle.

Conclusion

This paper explored the application of heuristic solutions for a real-world search problem, exemplified by the 8-puzzle. An interactive Python script, leveraging the A* search algorithm and the SimpleAI library, was presented to compute an effective solution based on a given 8-puzzle's initial state. The robustness of the script was showcased through its capability to detect user errors in input, such as unsolvable initial states or those in an incorrect format. The system is designed to alert users to such errors and subsequently prompt for correct input.

At the heart of this solution is the A* search algorithm. Its completeness and use of an admissible heuristic, specifically the Manhattan distance, ensure that the cost to reach the goal state is never overestimated. As such, the A* search emerges as cost-optimal, finding the solution with the lowest path-cost among all solutions, corresponding to identifying the solution with the fewest possible tile moves in the context of the 8-puzzle.

Furthermore, the evaluation function of the A* algorithm was explained, represented as $f(n) = g(n) + h(n)$. The key strengths and weaknesses of the algorithm were discussed, offering a balanced perspective on its utility. The comparison between the 8-puzzle problem and the concept of Multi-Agent Path Finding (MAPF) suggests potential real-world applications in domains requiring strategic planning and efficient resource management, such as traffic control, warehouse management, and video game design.

Given these characteristics, the A* search algorithm proves to be an efficacious mechanism for solving the 8-puzzle, contributing to the broader understanding of heuristic search algorithms and their potential in problem-solving within the field of artificial intelligence.

References

Collier, A. (2019, April 10). *Sliding Puzzle Solvable?* https://datawookie.dev/blog/2019/04/sliding-puzzle-solvable/

Felner, A., Stern, R., Shimony, S., Boyarski, E., Goldenberg, M., Sharon, G., ... & Surynek, P. (2017). Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges. In *Proceedings of the International Symposium on Combinatorial Search* (Vol. 8, No. 1, pp. 29-37).

Kunkle, D. R. (2001). Solving the 8-puzzle in a minimum number of moves: An application of the A* algorithm. *Introduction to Artificial Intelligence*.

Romanycia, M. H., & Pelletier, F. J. (1985). What is a heuristic? *Computational Intelligence*, *1*(1), 47–58.

Russell, S., & Norvig, P. (2002). *Artificial intelligence: A modern approach*.

Sharda, R., Delen, D., & Turban, E. (2020). *Analytics, data science, & artificial intelligence* (Eleventh edition). Pearson.

Welcome to simpleai's documentation! —Simpleai 0.8.2 documentation. (n.d.). Retrieved July 4, 2021, from https://simpleai.readthedocs.io/en/latest/