|  |
| --- |
| **Optimistic and Pessimistic Heuristics with Bounded Cost Mechanisms: A Comparative Study of A\* and RTA\* in the 8-Puzzle Problem** |
|  |
| **Robbes-Lasry Raphael, Sela Amitai**  Technion |
|  |
|  |
|  |
|  |

Abstract

While optimistic heuristics dominate traditional heuristic search due to their admissibility and widespread availability, recent studies [1] suggest that pessimistic heuristics may hold advantages in real-time, incomplete search contexts. This research builds on these insights, introducing a bounded-cost mechanism for dynamically managing heuristic selection in A\* and RTA\*. Our approach aims to do two things – limit the overuse of optimistic or pessimistic heuristics by incorporating a threshold-based switch to standard heuristic evaluation and check how well the pessimistic \ optimistic approach works with other heuristics. Using the 8-puzzle problem, we evaluate five heuristics across 30,000 puzzle instances with different levels of difficulty, analyzing solution length, node expansions, and decision accuracy.

Introduction

Based on the ideas discussed in Sadikov, A., & Bratko, I. (2006), we aim to further expand the applications of optimistic and pessimistic heuristics by integrating them into Real-Time Adaptive (RTA) search algorithms alongside A\* search. While the original paper applies these changes in heuristics to only the classic RTA\*, we introduce a variation by incorporating a *bounded cost mechanism*.

This bounded cost mechanism imposes a limit on the extent to which the algorithm can rely on the optimistic or pessimistic heuristic, before switching to the standard heuristic approach. The motivation for this modification is to mitigate potential drawbacks inherent to optimistic and pessimistic heuristics, such as overestimations or underestimations of the true cost to the goal.

In practice, the algorithm begins by using the optimistic/pessimistic heuristic – depending on the configuration – and once a cumulative cost exceeds a predefined threshold, the algorithm switches to the standard approach of calculating the heuristic for the rest of the computations.

The Problem

As said above, we are using the 8-puzzle sliding problem. The 8-puzzle is a simple known problem, where a 3X3 board is presented, with a number between 0-8 is in each tile – 0 represents the empty tile – and to solve the puzzle we need to slide the tiles to get it into a goal position, in this case to:

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 0 |

In our algorithm, a specific puzzle is represented by an *array* – for instance the array

represents the goal puzzle in our problem as shown above. We also have a handful of traits for easier algorithmic use, like puzzle parent, what type of heuristic we are using etc…

Notice that there are 188,140 possible solvable 8-puzzle variations, as shown [here](https://intellipaat.com/community/2366/how-many-possible-states-does-the-8-puzzle-have#:~:text=The%20feature%20of%20the%208,solved%20is%209!%2F2.).

Existing Work

To do this project, we relied on some previous work and research, as we see detailed below.

Algorithms:

The A\* algorithm, introduced by Hart et al. in 1968, uses the evaluation function

Where is the cost so far and is the estimated heuristic cost to the goal. A\* is widely used in heuristic search.

The RTA\* algorithm – in this algorithm, the merit of a node n is the same:

As in A\*, however, unlike A\*, the interpretation of g(n) in RTA\* is the actual distance of a node n from the current state of the problem solver, rather than from the original initial state – in our problem, this means that is always 1, since the distance between one problem to its neighbors is always one move.

The key difference between RTA\* and A\* is that in RTA\*, the merit of every node is measured relative to the current position of the problem solver, and the initial state is irrelevant. RTA\* is simply A best-first search given this slightly different cost function.

Regular Heuristics

In this project we used 4 main heuristics and h\* to test our algorithm:

* ***The Manhattan Distance*** – sum of the Manhattan distances of tiles that are not in their correct place.
* ***Misplaced tiles*** – the number of tiles that are not in their goal position.
* ***Gaschnig relaxed adjacency*** - The number of tiles that need to be moved to reach the goal state, assuming any tile can be swapped with the empty tile.
* ***Linear conflict*** – Tiles are in linear conflict if they are in the same row or column, their goal positions are also in the same line, tile 1 is to the right of tile 2, and the goal position of tile 1 is to the left of tile 2. To calculate the heuristic value, determine the number of tiles with a linear conflict in each row, subtract 1 for each row, and do the same for each column. Add all these values, multiply the result by 2, and then add the Manhattan distance to obtain the final heuristic value.

We are also using the h\* heuristic – meaning the real distance between the puzzle and the goal state.

To calculate each special heuristic, we coded a function that directly calculates it, and to calculate h\*, we run beforehand a BFS starting from the goal state, and saved the distances in a Json file, not to repeat the heavy calculation each time.

New Heuristics usage

As shown in Sadikov, A., & Bratko, I. (2006), we used the optimistic pessimistic approach to heuristics – in short, this basically means adding – or subtracting it – a random noise to the final heuristic value, using the heuristic value to make the noise. In our project we made the noise in the same way the paper did – a gaussian distribution, with a fixed standard deviation, and the actual heuristic value as the mean.

Our algorithm

In our algorithm, we selected a sample of 30,000 puzzles from the 181,440 possible solvable 8-puzzle configurations. This decision was primarily influenced by hardware limitations and time constraints. Running the algorithm on the entire dataset for each iteration was not possible, as each puzzle needed to be evaluated using five heuristics, using optimistic, pessimistic, and standard heuristic calculation, and with and without cutoff.

The computational demands were large, as even processing 30,000 puzzles required approximately 12 to 16 hours per run on our personal computers. Scaling up to include all 181,440 puzzles would have extended the runtime to several days, which was beyond our available resources and practical timelines.

To gain meaningful insights into the algorithm’s performance, we divided all puzzles into three distinct groups based on their distance from the goal state: *starting puzzles* (close to the goal), *middle puzzles* (moderately distant), and *end puzzles* (far from the goal). From each group, we randomly selected 10,000 puzzles, resulting in a balanced dataset with an equal representation of easy puzzles near the goal, intermediary puzzles at moderate distances, and challenging puzzles far from the goal. This approach ensured that our sample reflected the full range of puzzle difficulties, allowing us to evaluate the algorithm's performance comprehensively across different levels of complexity.

Explanation

Our model is comprised of 2 phases:

* *Phase 1:*

We run the wanted algorithm, either A\* or RTA\*, with the specific way we want to calculate the heuristic – either optimistic or pessimistic and count the number of nodes we explore.

When this number crosses a *threshold*, we go into phase 2

* *Phase 2:*

Run the rest of the algorithm, just with standard heuristic calculation.

The algorithm is of course guaranteed to find eventually the solution, since it will either find a solution using the optimistic / pessimistic approach or expand enough nodes to just run A\* or RTA\*, that are guaranteed to find a solution.

* 1. Parameters

We run our model on the following parameters:

* Heuristics:
  + A\*
  + RTA\*
* Heuristics:
  + h\*
  + Manhattan distance
  + Linear Conflict
  + Misplaced tiles
  + Gaschnig relaxed adjacency
* Heuristic calculations
  + Optimistic
  + Pessimistic
  + Basic (plain heuristic value)
* With \ without cutoff of:
  + 45360, 22680, no cutoff

*The cutoff values are the number of possible, solvable 8-puzzles divided by 4 and 8.*

Empirical Results

Evaluation

To evaluate our algorithm performance, we used 4 metrics:

* Time

For each puzzle, how much time has it taken to solve. Low time is better.

* Nodes expanded

How many nodes did the algorithm explore in his search. Low number of nodes is better.

* Solution length:

How long the final path that was given is – the longer the path. Low solution length is better.

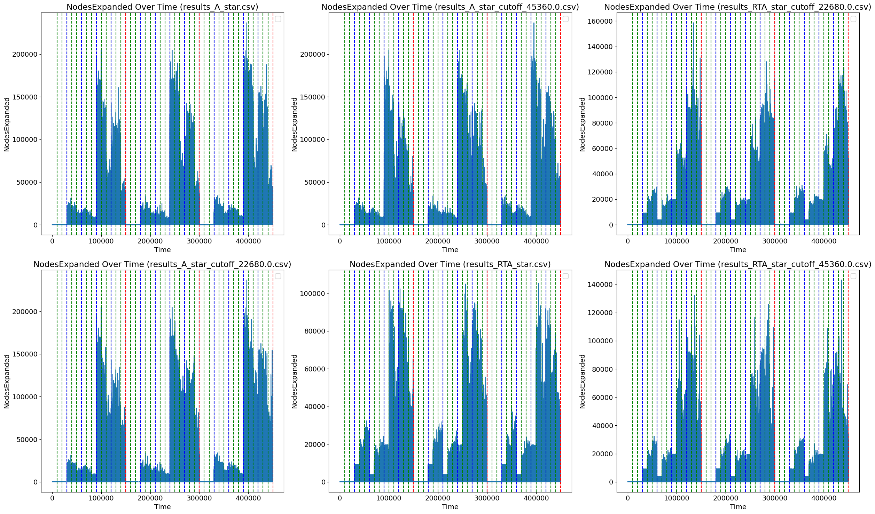
* Right decisions

The percentage of right decisions in the path, relative to the whole path.

*A right decision* is measured using h\* – if the actual distance of a puzzle in the final path is higher than the next puzzle in the path, that means we are going in the right direction, and a right decision was made.

High right decision is better.

We calculated and saved these metrics in our csv’s. Since we run this on three relatively old computers, we are going to focus mainly on the *expanded nodes*, *right decision* and solution length metrics, since we can’t rely on comparing the times.

Raw Results

(Better resolution image is in the [appendix](#Expand) and in our [git repository](#gitamit)).

The graphs above depict the progression of nodes expanded over time during the search. Here's how to interpret the visual elements:

* **Red vertical lines**: Indicate a transition to a different puzzle state, progressing through difficulty levels from "easy" to "harder" as described earlier.
* **Blue vertical lines**: Mark a change in the heuristic being used. The heuristics follow this sequence:
  1. h\*
  2. Manhattan Distance
  3. Linear Conflict
  4. Misplaced Tiles
  5. Gaschnig's Relaxed Adjacency
* **Green vertical lines**: Represent a change in the calculation approach. The order is as follows:
  1. Basic
  2. Optimistic
  3. Pessimistic

Each graph’s title contains the corresponding file name and the cutoff value used. Graphs without a cutoff value indicate no restrictions were applied during the search. Additional graphs can be found in the appendix, and higher-resolution versions are available in the code.

* 1. Comparing Heuristics

For the number of nodes expanded, the results were split between A\* and RTA\*. In the case of A\*, both the mean and median values for each heuristic were lower when using the pessimistic approach compared to the optimistic approach, with the exception of the Manhattan distance heuristic, where the median was better under the basic approach.

In contrast, RTA\* exhibited different patterns. For all heuristics except Gaschnig’s Relaxed Adjacency, the basic approach consistently outperformed the pessimistic approach in terms of both mean and median values. Notably, for the Manhattan distance heuristic, the optimistic approach yielded a better median result.

Solution length also demonstrated a clear divide between A\* and RTA\*. A\* with the basic approach consistently achieved better or same solution lengths across all heuristics. However, in RTA\*, the pessimistic approach produced the best mean solution length for most heuristics, with the exception of h\*, where the basic approach was optimal. The median solution length was similarly dominated by the pessimistic approach, except for h\* and the misplaced tiles heuristic.

For the right decisions, in A\*, the basic approach consistently achieved better mean right decisions across all heuristics, with no scenarios where either the pessimistic or optimistic approaches outperformed it. Median results further emphasized this trend, with the basic approach outperforming the other strategies for Manhattan distance and linear conflict heuristics, while achieving parity with the pessimistic approach for h\*, misplaced tiles, and Gaschnig’s Relaxed Adjacency heuristics.

In RTA\* presented a different narrative. For most heuristics, including Manhattan distance, linear conflict, misplaced tiles, and Gaschnig’s Relaxed Adjacency, the pessimistic approach demonstrated superior performance in terms of mean right decisions. The basic approach only outperformed the pessimistic approach for h\* in this category. Median results followed a similar trend, with the pessimistic approach achieving the best outcomes for most heuristics, except for misplaced tiles, where the basic approach prevailed, and h\*, where the results were comparable between the two strategies.

* 1. Comparing cutoffs

We see that the cutoff benefited the A\* algorithm, where both a cutoff of 22,680 and 45360 gave better results in nodes expanded, in terms of mean – where in terms of median, the results where almost identical.

In RTA\* the cutoff worsens the performance, but again in terms of average, not in terms of median.

* 1. Comparing algorithms

In terms of nodes expanded, A\* did measurably worse than RTA\* in terms of average, but in terms of median did better. In solution length, RTA was demolished by A\*, giving averages of around 350, and medians of around 196, while A\* gave averages and medians of around 25.

In terms of right decisions, again A\* demolishes, giving around 0.965 averages and ~1.0 median, while RTA\* struggles with ~0.65 averages and 0.55 medians.

Conclusion

* 1. Main conclusions

Our findings demonstrate that the choice of heuristic and its calculation approach significantly affects the performance of A\* and RTA\*. While A\* generally outperformed RTA\* in terms of solution length and decision accuracy, RTA\* expanded fewer nodes on average, making it potentially more suitable for resource-constrained applications.

Additionally, the pessimistic heuristic approach consistently improved results in RTA\* across most metrics, indicating its potential for real-time decision-making tasks, and confirming with the results of Sadikov, A., & Bratko, I. (2006), the basic heuristic approach proved more reliable for A\*, particularly for A\*, and the optimistic approach showed almost no potential.

* 1. The mean-median phenomenon

We observed a lot of differences between mean and median metrics. For instance, A\* showed higher average node expansions compared both to RTA\* and compared to itself, but this difference was not reflected in the median, suggesting that it’s prone to sometime expand in extreme ways, skewing the mean.

Limitation

Our main limitation is time and computing power. We would have loved to have a more powerful machine to run the algorithm faster, with more data and on more cutoff values.

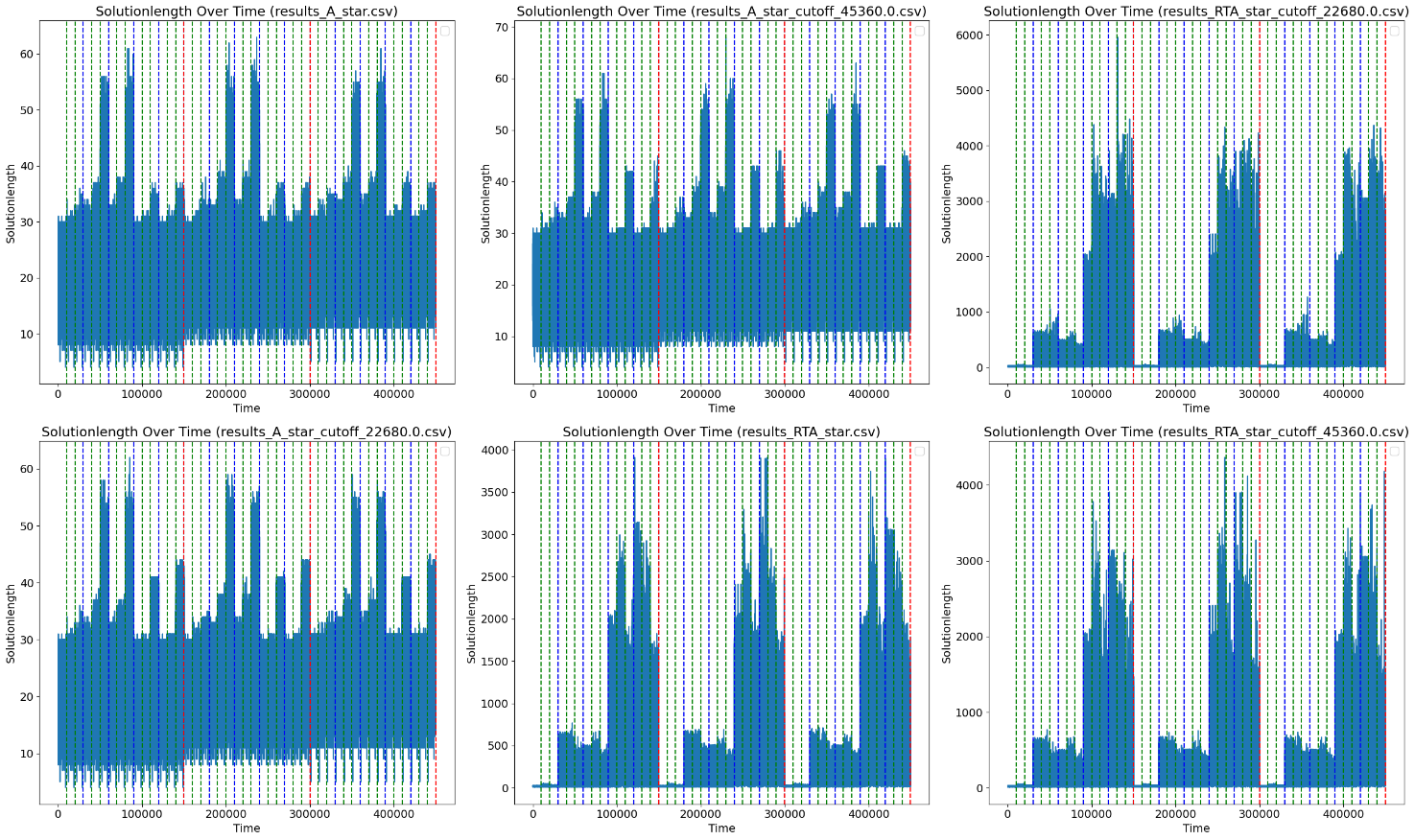
Another enhancement we could have explored, given more time, would be experimenting with alternative methods for calculating the optimistic and pessimistic heuristics, specifically by varying the *c* values and the *SD* values.

References

[1] Sadikov, A., & Bratko, I. (2006). Pessimistic heuristics beat optimistic ones in real-time search. Frontiers in Artificial Intelligence and Applications, 141, 148.

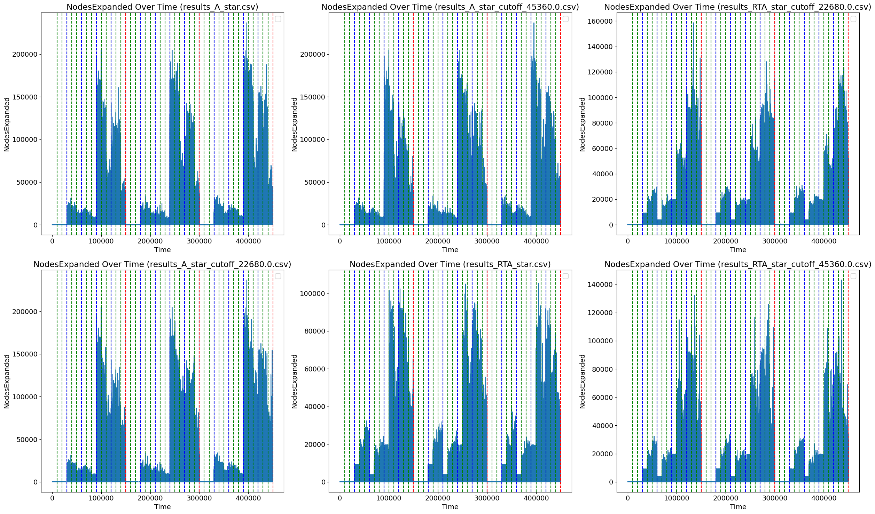
<https://github.com/Amitaisela/BINA_Project>

1. Appendices

Solution length over time:

A group of blue and green vertical lines

Description automatically generatedRight decisions over time:

Nodes expanded over time:

[To return to the paper](#OLE_LINK34)