A Neural Network Case Base for Real-Time Searching

Zeyi Wang
Department of Computing Science
University of Alberta
Edmonton, Alberta, T6G 2E8, Canada
zeyi2@ualberta.ca

(When I use "I" instead of "we", that means that part is my personal reflections and will changed in the final report.)

Abstract

Some real-time search algorithms ~~use~~ a case base to store previously solved problems and use the case base to build a new solution for an unknown problem in the real-time. Larger case bases leads to better performance at the cost of ~~slower~~ longer query time. In this paper, we build a neural network that is trained on the cases and achieve constant time for queries in any size case bases.

1 Introduction

Using a case base is common technique in real-time search algorithms. When an agent encounters a previously unknown task, it searches in its case base for similar situations, and infers a solution from the cases. As a case base covers more cases, an agent utilizing the case base will be more likely to find similar situations and solutions. However, the time cost of retrieving similar cases also goes up when the case base is larger. In this paper, we purpose Neural Network Case Base (NNCB), a case base that benefits from more cases, but does not suffer from the extra time cost.

Right now I have a few ideas of doing this.

- build a data base for inferring amount of heuristic updates
- build a data base for inferring actions
- build a data base for inferring an optimal path

2 Problem Formulation

We define a heuristic problem as a directed graph containing a finitely many number of *states*, in which one is the *start* and one is the *goal*, and *edges* connecting adjacent states with fixed costs of 1. The agent has a single *current state* at every time step. The agent can take an action by following an edge to change its current state. The *solution cost* is the sum of all edge costs of a path the agent can follow to go from the start state to the goal state. The algorithm has to be complete so it is comparable to ~~the~~ other algorithms.

3 Related Work

So far I have been comparing my algorithm with weighted LRTA* by Bulitko & Lee (2006) Rivera, Baier, & Hernández (2015). I implemented the neural network using pytorch by Paszke *et al.* (2019). One of the approaches I am going to try involve predicting the amount of learing and there's previous work on that by Lelis *et al.* (2012). I drew inspiration from D LRTA* by Bulitko *et al.* (2007), kNN LRTA* by Bulitko & Björnsson (2009) and HCDPS by Lawrence & Bulitko (2010) for using a case base. I also drew inspiration from RTNN by Muñoz *et al.* (2018) on using a neural network on real-time search problems.

(I am not sure how to cite these properly yet, but I will figure this out before the final report.)

4 Proposed Approach

For this midterm report, most of my effort has been coding infrastructures (e.g, map/scene parsing, map/scene/path/heuristic-updates visualizations, agent-map interaction framework). See Figure 1 and Figure 2 for visualizations.

The first thing I tried is to build a case base to predict heuristic updates from a view and its distance to the goal.

Formally, for each entry in the case base, we have:

- $V_{i,j}$: a 7x7 matrix around the current state
- $d_{i,j}$: a 2D vector containing the difference between the goal and the current state
- $h_{i,j}$: the difference between the current state and the goal state

To use the case base, we start with a standard LRTA* procedure. When the agent wants to update

Figure 1: LRTA* Planned Path Visualized

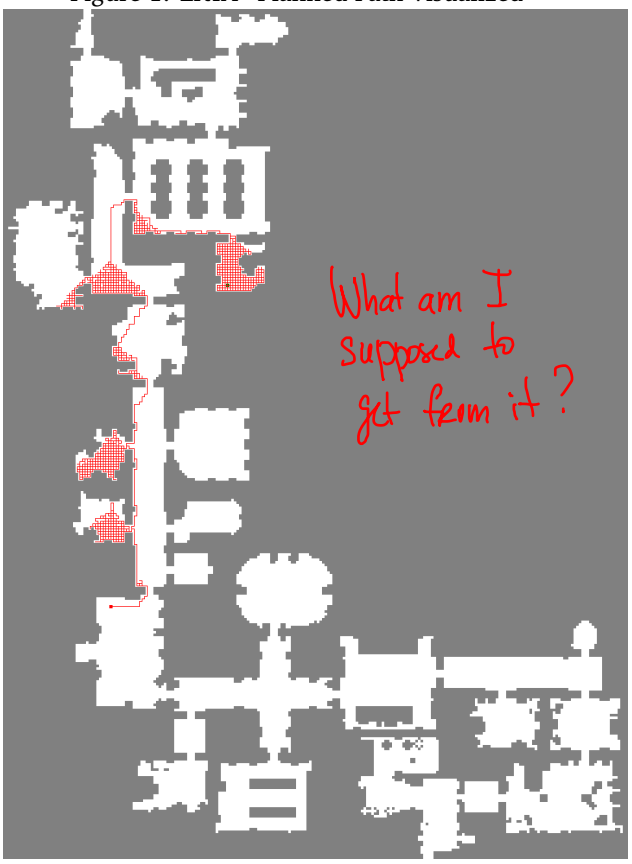
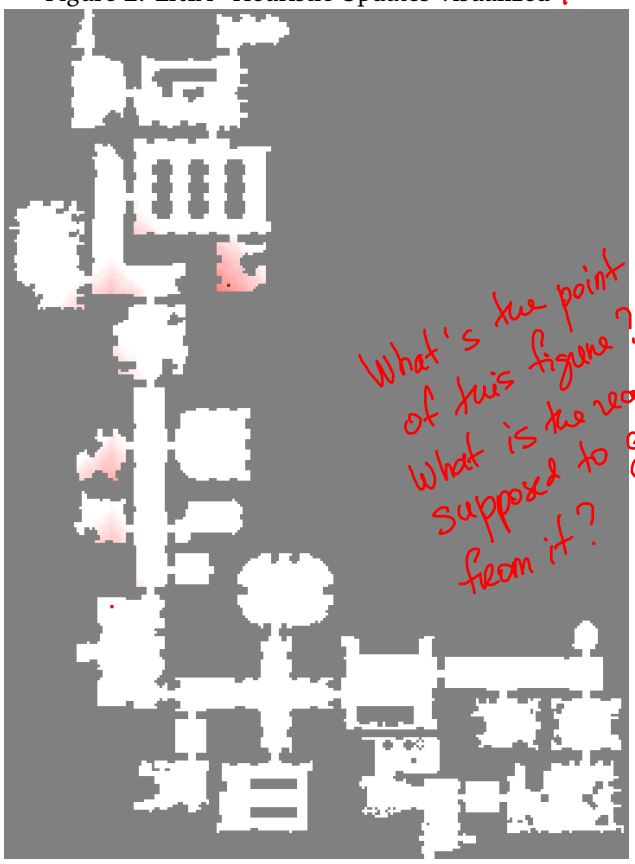


Figure 2: LRTA* Heuristic Updates Visualized



Do not use caps on each word.

a heuristic at given position i, j, instead of update the heuristic with a new f, the agent query the case base with $V_{i,j}$ and $d_{i,j}$. Right now my program does this by finding the nearest neighbors in the case base and averaging them. However, this part will be replaced by a neural network later.

(I also want to try predicting a move instead, we will see.)

how many?

5 Theoretical Analysis

5.1 Completeness

We keep track of the number of visitations of each state. Once the number of visitation of a certain state exceeds M, a constant that we select, then the agent plan with vanilla LRTA* on that state. Since LRTA* is complete, and our algorithm at most plans $M * number of states$ times, the whole algorithm is complete.

I do not know how you average cases. If you mean that you average h values associated w/ each case then you need to say so and give a formulation for weighted (?) average.

5.2 Offline Time Complexity

The time is linear to the number of cases generated.

to do what?

5.3 Offline Space Complexity

Since we are training a neural network with cases, we don't need to store the cases explicitly and the cases can be discarded once the network is trained. As a result, the offline space required by the algorithm is purely based on the number of parameters of the network, which is a constant.

relative to? All complexity analysis should refer to size of the problem in some way.

5.4 Online Time Complexity

Once we train a neural network for the case base, each query takes a fixed amount of computational time. Since the LRTA* backbone of the algorithm also runs in constant time, the whole algorithm runs in constant time.

Not relative to map size.

5.5 Online Space Complexity

The agent only store sub-goals in its memory and the number of sub-goals is upper-bounded by the number of states in the map. Hence, the online space complexity is a constant.

what subgoals?

relative to what?

Only on safely explorable, finite graphs. You had no such restriction in Section 2.

And what about training the network?

6 Empirical Evaluation

Evaluate on *Dragon Age: Origins* (DAO) maps provided by Sturtevant (2012). So far I evaluated the draft version (not using a neural network) on one map and compared it with weight LRTA* ($w = 4$), and their suboptimality is roughly the same. I am planning to evaluate my algorithm on more DAO maps. In addition, I would like to split the pool of maps into a *training pool* and a *testing pool*. I will try to build the case base on the training pool and see if the algorithm works well on the testing pool.

7 Discussion

We think using a neural network to build a case base is powerful idea and has a ~~huge~~ potential. We can discard the collected cases after the training is done (or even discard them as we train) to achieve a constant offline space complexity. We can also have constant time look up in the case base no matter how many cases we collected. We can think of a neural network to be the ~~ultimate~~ way to compress information in a case base to a fixed size.

8 Future Work

One thing we would like to try is to create a neural network powered case base for querying optimal paths. The input of the network will be the start position and the goal position. The output of the network will be a sequence of hill-climbable sub-goals. Online, the agent retrieves the sub-goals and follows along by hill-climbing. Algorithms like kNN LRTA* suffers from case base offline space complexity and online time complexity. However, if we can train our case base with infinitely many cases without worrying about the consequences, we can make that infinite knowledge shines.

9 Conclusions

I expect to draw a conclusion based on my empirical evaluation.

Acknowledgments

I gratefully acknowledge the help by Vadim Bulitko, without which the present study could not have been completed.

References

Bulitko, V., and Björnsson, Y. 2009. knn lrta*: Simple subgoaling for real-time search.

Bulitko, V., and Lee, G. 2006. Learning in real-time search: A unifying framework. *Journal of Artificial Intelligence Research* 25:119–157.

Bulitko, V.; Björnsson, Y.; Schaeffer, J.; and Sigmundarson, S. 2007. Dynamic control in path-planning with real-time heuristic search.

Lawrence, R., and Bulitko, V. 2010. Taking learning out of real-time heuristic search for video-game pathfinding. In *Australasian Joint Conference on Artificial Intelligence*, 405–414. Springer.

Lelis, L. H.; Arfaee, S. J.; Zilles, S.; and Holte, R. C. 2012. Learning heuristic functions faster by using predicted solution costs. In *SOCS*.

Muñoz, F.; Fadic, M.; Hernández, C.; and Baier, J. A. 2018. A neural network for decision making in real-time heuristic search. In *Eleventh Annual Symposium on Combinatorial Search*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 8024–8035.

Rivera, N.; Baier, J. A.; and Hernández, C. 2015. Incorporating weights into real-time heuristic search. *Artificial Intelligence* 225:1–23.

Sturtevant, N. 2012. Benchmarks for grid-based pathfinding. *Transactions on Computational Intelligence and AI in Games* 4(2):144 – 148.

an incomplete sentence

But you are eliminating the case base! So there is no look up.

You are not "powering" (whatever that means) a case base. You are removing a case base!

I do not understand this. As far as I can tell you are replacing a case base w/ a neural network. If so then you are not using the network to build a case base!

Space complexity / memory was not defined / discussed in problem formulation. So why do we care about it?

wrong case

where published?

where published?

Proceedings of

expand. Give pages.

Proceedings of

Paszke