CPS8326 Paper Reflection 1: Towards Rational Deployment of Multiple Heuristics in A*

Colby Ziyu Wang

September 2024

1 Paper Reflection

In the paper titled *Towards Rational Deployment of Multiple Heuristics in A**, the authors address the efficiency problem encountered by the A^*_{MAX} algorithm when dealing with multiple admissible heuristics. Specifically, this occurs when the maximum of the heuristics is used as the admissible heuristic for each generated node. Although this approach ensures optimality, it is inefficient because all heuristics must be computed for every node, leading to unnecessary computational overhead.

The main contributions of this paper are as follows: the first is a proposed variant of the A^*_{MAX} algorithm, namely Lazy $A*$. In this algorithm, we assume that there are two admissible heuristics h_1 and h_2, where $h_1(n) \leq h_2(n)$ for most nodes n. We also assume that the computation time of h_2 is much greater than that of h_1. The main difference between Lazy $A*$ and A^*_{MAX} is that when a node n is generated, only h_1 is computed. After n is first removed from OPEN, h_2 is computed and the node is reinserted into OPEN with $f_{max}(n)$.

There are two enhancements that can be made to Lazy $A*$, called OPEN bypassing and heuristic bypassing. The OPEN bypass works when $h_1(n)$ is computed and $h_1(n) \leq f_{best}$. In this case, n will reach the top of OPEN immediately and $h_2(n)$ can be computed right away, saving the overhead of inserting n into OPEN and popping it again at the next step. Heuristic bypassing is a technique that allows A^*_{MAX} to bypass evaluating one of the heuristics. Here, it is assumed that the operator between a node n and its parent p is bidirectional, and both h_1 and h_2 are consistent. This implies $|h(p) - h(n)| \leq C$, and $h(p) - C \leq h(n) \leq h(p) + C$. Denote the lower bound as $\underline{h(n)} = h(p) - C$ and the upper bound as $\overline{h(n)} = h(p) + C$. When $\underline{h_1(n)} < \overline{h_2(n)}$, the computation of h_2 is delayed and n is added to OPEN using $\overline{g(n) + h_2(n)}$.

The second main contribution made by the authors is the Rational Lazy $A*$ algorithm. This algorithm seeks to manage the trade-off between expanding more states to reduce search time. When $h_2(n)$ is computed but the goal is found without expanding n, the time wasted is denoted as t_d. If $h_2(n)$ is not computed but would have been helpful, the time wasted is $t_e + b(n)t_d$, where t_e is

the time for removing n from OPEN, expanding n, evaluating h_1 on each of the $b(n)$ ("local branching factor") children n' of n, and inserting n' into the open list. Computing h_2 would have cost t_d, so the regret here is $t_e + (b(n) - 1)t_d$.

Now, denote p_h as the probability that h_2 is helpful. The heuristic is computed when:

$$(1 - p_h)t_d < p_h \left(t_e + (b(n) - 1)t_d\right)$$

to minimize regret. Thus, the decision is to evaluate h_2 when:

$$t_d < \frac{p_h}{1 - p_h b(n)} t_e$$

for $p_h b(n) < 1$.

To evaluate the proposed algorithms, the authors utilized the weighted 15-puzzle and Planning domains. In the case of the weighted 15-puzzle, the authors used the Weighted Manhattan heuristic (WMD) as h_1, and an expensive heuristic based on lookaheads as h_2. They reported the runtime of A^*_{MAX}, Lazy $A*$, and Rational Lazy $A*$, as well as metrics such as the number of nodes saved from computing h_2 and how often h_2 is computed in LA* and RLA*. The authors observed that RLA* performed far fewer h_2 computations, resulting in decreased runtimes.

In the second empirical evaluation, the authors implemented LA* and RLA* on the Fast Downward planning system. They compared the performance of LA* and RLA* to that of A* using each heuristic individually, as well as to their max-based combination and selective-max (SelMAX). The authors found that RLA* solved the most problems and was the fastest overall, while LA* solved the same number of problems as SelMAX.

The limitations of LA* and RLA* are demonstrated in the case of the regular 15-puzzle problem, where either h_1 has a similar cost to h_2 or h_2 dominates h_1. The savings in these cases are not significant. The authors conclude that LA* is likely to be effective when there is a significant time difference between computing h_1 and h_2, or when operators are not bidirectional and/or have non-uniform costs, allowing for greater time savings.

One way to build on this paper is to test the algorithm on more recent datasets, as the dataset used for planning domain is from 2006. For example, applying Rational Lazy A* to contemporary challenges like autonomous driving, where real-time decision-making is critical, could provide deeper insights into its practical relevance.

Two final questions arise: First, why is the maximum of two admissible heuristics still admissible? To show this, consider two admissible heuristics h_1 and h_2, both of which are less than or equal to $h^*(n)$. Since $\max(h_1, h_2) \leq h_1$ and $\max(h_1, h_2) \leq h_2$, by transitivity we obtain $\max(h_1, h_2) \leq h^*(n)$.

The second question concerns the worst case for Rational Lazy A*. The worst case occurs when the computation times for h_1 and h_2 are very similar and very small, or when h_2 dominates h_1. In such cases, the time savings are minimal, and h_2 is required early (e.g., by Open Bypassing), while h_1 rarely influences the search.