Anytime Heuristic Search

October 6, 2024

The problem this paper aims to address is the challenge of balancing the trade-off between search time and solution quality in complex search problems, particularly when the available time is uncertain. Specifically, the paper explores the scenario where an initial solution is found quickly using a non-admissible heuristic, but further search continues to improve the solution until the given time or computational resources are exhausted. The focus is on developing an effective strategy that provides a suboptimal solution rapidly while enabling iterative refinement toward optimality, based on the time available.

The main contribution of this paper is the proposed Anytime Weighted A* (Anytime WA*) algorithm. This algorithm leverages two heuristic functions. The first, denoted $h'(n)$, is a non-admissible weighted heuristic (i.e., $w \cdot h(n)$, where $h(n)$ is admissible), used to select nodes for expansion. This allows the algorithm to find good but possibly suboptimal solutions quickly. The second heuristic is an admissible heuristic, $h(n)$, which is combined with an upper bound on the optimal solution cost (the cost of the best solution found so far) to prune the search space and detect convergence. Specifically, pruning occurs when the sequence of improved solutions provides upper bounds on the optimal solution cost, enabling Anytime A* to compare the f-cost of each successor of the current node against the upper bound. If the f-cost exceeds the upper bound, the successor is not added to the OPEN list, as it cannot lead to a better solution. Additionally, the paper proves that the algorithm always terminates and converges to the optimal solution. It also establishes an error bound: $\frac{f(\text{incumbent})}{f^*} \leq \frac{f(\text{incumbent})}{f_L}$ where f_L is the lower bound on the optimal solution cost, $f(\text{incumbent})$ is the upper bound, and f^* is the optimal solution cost. In simpler terms, the least f-cost of any currently open node provides a lower bound on the optimal solution cost, while the upper bound is given by the cost of the best solution found so far.

To evaluate the proposed Anytime WA* algorithm, the authors first applied it to the sliding-tile puzzle problem. They considered eight such instances and compared the solution quality, defined as $1 - \frac{f-f^*}{f^*}$, for weight values $w = 1.3$, $w = 1.5$, and $w = 2.0$. The results showed that solution quality improved as more computation time was allowed, with the weight $w = 1.3$ providing the best overall performance. Furthermore, the authors observed that the average increase in the number of nodes expanded by Anytime WA* using $w = 1.3$ was minimal compared to unweighted A*. Additionally, Anytime WA* with weights

$w = 1.3$ and $w = 1.5$ expanded fewer nodes on average than unweighted A*. The second domain the authors tested is domain-independent STRIPS planning. They employed the max-pair heuristic proposed by Haslum and Geffner (2000) with a weight of $w = 2.0$. The observations show that Anytime WA* consumed less memory than A* and was faster than A* in most of the tested instances. The authors also introduced a percentage metric that measures the number of node expansions required to find the best solution relative to the total number of node expansions until convergence. In most domains, Anytime WA* quickly finds the optimal solution and spends the remaining time verifying that the solution is indeed optimal. The third domain explored by the authors is the problem of aligning DNA sequences, which can be modeled as a shortest-path problem in an n-dimensional lattice, where n represents the number of sequences to be aligned. The authors compared Anytime WA* with A*, A* with Partial Expansion (PEA*)—which inserts only the most promising successor into the OPEN list—and Enhanced A* (EA*)—which uses the upper bound obtained by a solution with $w > 1$. Their results indicate that all modified A* algorithms consumed less memory than standard A*. Moreover, Anytime WA* outperformed PEA* and EA* in terms of runtime and number of nodes stored. This improvement is due to Anytime WA* finding a suboptimal solution quickly and continuously refining it with additional computation.

In terms of future work, one interesting direction could be to investigate the effect of dynamically adjusting the weight w during the execution of Anytime WA*, such as gradually lowering the weight as the search progresses. This approach may help the algorithm converge faster to the optimal solution, as a lower w value typically provides better heuristic estimates, allowing the search to focus more on optimality rather than speed once a suboptimal solution has been found.

Finally, two key questions arise. The first concerns the use of a heuristic that may not be guaranteed to be admissible, but is necessary to guide the search towards an optimal solution in Anytime Heuristic Search. The proposed solution is as follows: for the admissible evaluation function in Anytime A*, we use Dijkstra's algorithm. For the non-admissible evaluation function, we define $f'(n) = g(n)+w\times h_1(n)$, where $h_1(n)$ is the heuristic that may not be admissible. The rationale behind this is that if $h_1(n)$ is admissible, then $w\times h_1(n)$ becomes a weighted heuristic (which satisfies the requirements of Anytime WA*); if $h_1(n)$ is not admissible, then $w \times h_1(n)$ acts as a non-admissible heuristic, which satisfies the requirements of Anytime A* (since Anytime WA* can be viewed as a special case of Anytime A*). The second question concerns why Anytime WA* sometimes finds optimal solutions in less runtime compared to A*, even though it expands more nodes. The reason is that WA* uses a criterion for inserting the successors of a node into the OPEN list, which is guided by the upper bound provided by the best solution found so far. This pruning mechanism allows Anytime WA* to discard nodes that cannot lead to better solutions, resulting in savings in time and memory. These savings sometimes outweigh the overhead incurred by expanding more nodes.