Proceedings of the Twelfth International
Symposium on Combinatorial Search (SoCS 2019)

A-MHA*: Anytime Multi-Heuristic A*

Ramkumar Natarajan,[†*] Muhammad Suhail Saleem,[†*] William Xiao,[†]
Sandip Aine,[‡] Howie Choset,[†] Maxim Likhachev [†]
[†]The Robotics Institute, Carnegie Mellon University
[‡] Apple Inc.

Introduction

While designing a single heuristic function that guides the search well is challenging, it has been shown that multiple heuristics can often dramatically speed up the search (Helmert 2006). In fact, it is often easy to design heuristics that perform well and correlate with the underlying true cost-to-go values in *certain* parts of the search space but these may not be admissible throughout the domain thereby affecting the optimality guarantees of the search. Bounded suboptimal and complete search using several such partially good but inadmissible heuristics was developed in Multi-Heuristic A* (MHA*) (Aine et al. 2016). Although MHA* leverages multiple inadmissible heuristics to potentially generate a faster suboptimal solution, the original version does not improve the solution over time.

Real world and real-time planning on the other hand often need to trade-off solution quality for runtime. To that end, anytime algorithms have been developed that can generate a quick suboptimal solution and keep improving it over time. In this work, we extend MHA* to an anytime version by borrowing some of the concepts from Anytime Repairing A* (ARA*) (Likhachev, Gordon, and Thrun 2004) that runs a series of Weighted A* (WA*) (Pohl 1970) searches, each with a decreasing weight on heuristics. We ensure that our precise adaptation of ARA* concepts in the MHA* framework preserves the original suboptimality and completeness guarantees and enhances MHA* to perform in an anytime fashion.

Anytime Multi-Heuristic A* (A-MHA*)

Notations: Let $s \in \mathcal{S}$ denote the finite set of discrete states over which we search for a path from s_{start} to s_{goal}. The search typically proceeds by expanding states to generate successors $s' \in Succ(s)$ based on a priority. The current best cost and the optimal cost to arrive at a state s is denoted by $g(s)$ and $g^*(s)$. $c(s, s')$ denotes the cost between any two states s and s' connected by an edge. MHA* incorporates a single admissible heuristic $h_0(s)$ and multiple inadmissible heuristics denoted by $h_i(s),\ i = 1, ..., N$. Let the inflation of the searches be w_1 and let w_2 be the inflation factor to prioritize inadmissible search. Because of the anytime nature of the algorithm, the inflation factors are updated and the found solution is improved over time. With one admissible heuristic and N inadmissible heuristics, the $N + 1$ priority queues of expansion are given by $OPEN_0$ and $OPEN_i,\ i = 1, ..., N$ respectively. The priority of the states in $OPEN_i$ and $OPEN_0$ are given by $key(s, i) = g(s) + w_1 * h_i(s)$. In order to track and prevent re-expansions within a single search improvement routine, we have anchor and inadmissible closed lists and an inconsistent list denoted as $CLOSED_{anch}$, $CLOSED_{inad}$, and $INCONS$ similar to the lists used by ARA*.

Algorithm

The pseudocode of the proposed algorithm is presented in Algorithm 1. The structure of A-MHA* is similar to ARA*. The MAIN() function consists of the outer loop from which the IMPROVEPATH() function is called with the updated suboptimality bound. IMPROVEPATH() function is a modified MHA* routine that guarantees $w_1 * w_2$ suboptimality and keeps track of inconsistent states to reuse the search results during the next iteration.

Properties of A-MHA*

We present two important properties of A-MHA*: (1) The solution provided by any IMPROVEPATH() call has cost that is at most $w_2 * w_1$ suboptimal (*i.e*, no worse than $w_2 * w_1$ times the cost of an optimal solution), which extends from the detailed proof presented in the original MHA* paper (Aine et al. 2016). (2) Within each call of IMPROVEPATH(), a state is expanded at most twice. This follows from the fact that a state expanded by a call to EXPAND(S) from anchor search can never be expanded again, whereas a state expanded by a call to EXPAND(S) from an inadmissible search can be expanded only by the anchor search.

Experimental Setup, Results and Discussion

We evaluate the performance of A-MHA* on the sliding tiles puzzle and 3D navigation (x,y,orientation) domains. We include MHA*, ARA* and Anytime Nonparametric A* (ANA*) (Van Den Berg et al. 2011) in our comparisons.

[*]These authors contributed equally.

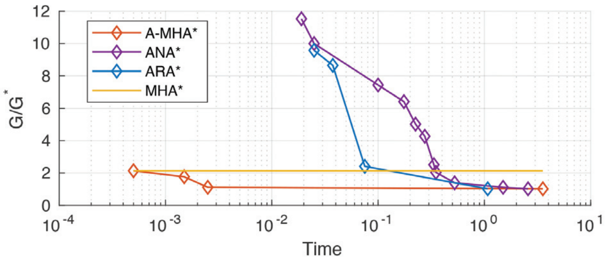

Figure 1: Average suboptimality of solution vs time for 50 instances of 3D planning experiments, Note that the time axis is in log-scale.

Algorithm 1 Anytime Multi Heuristic A* algorithm

procedure KEY(s, i)
  return $g(s) + w_1 * h_i(s)$;
procedure EXPAND(s, i)
  Remove s from $OPEN_i \; \forall \; i = 0, 1...N$
  for each s' in Succ(s)
   if $g(s') > g(s) + c(s, s')$
    $g(s') = g(s) + c(s, s')$
    if s' in $CLOSED_{anch}$
     Add s' to $INCONS$
    else
     Insert/Update s' in $OPEN_0$ with KEY$(s', 0)$
     if s' not in $CLOSED_{inad}$
      for $i = 1$ to n
       if KEY$(s', i) \leq w_2 *$ KEY$(s', 0)$
        Insert/Update s' in $OPEN_i$ with KEY(s', i)
procedure IMPROVEPATH()
  while $f(s_{goal}) > w_2 * OPEN_0.Min()$
   for $i = 1...N$
    if$(OPEN_i.Min() \leq w_2 * OPEN_0.Min())$
     s = $OPEN_i.Top()$
     EXPAND(s, i) and Insert s in $CLOSED_{inad}$
    else
     s = $OPEN_0.Top()$
     EXPAND$(s, 0)$ and Insert s in $CLOSED_{anch}$
procedure MAIN()
  $w_1 = w_1^0$; $w_2 = w_2^0$; $g(s_{start}) = 0$; $g(s_{goal}) = \infty$;
  for $i = 0...N$
   $OPEN_i$ = NULL
   Insert s_{start} in $OPEN_i$ with KEY(s, i)
   while $w_1 \geq 1$ and $w_2 \geq 1$
    $CLOSED_{anch}$ = $CLOSED_{inad}$ = NULL
    $INCONS$ = NULL
    IMPROVEPATH()
    Publish current $w_1 * w_2$ suboptimal solution
    if $w_1 == 1$ and $w_2 == 1$
     return
    $w_i = max(w_i - \Delta w_i, 1)$; $i = 1, 2$
    Move states from $INCONS$ into $OPEN_0$
    Copy all states from $OPEN_0$ to $OPEN_i$
    Update the priorities $\forall s \in OPEN_i; \forall i = 0..N$

3D Path Planning: We plan for a polygonal robot with three degrees of freedom (x,y,orientation) and minimum turning radius constraints in a 2-D planar environment. The consistent heuristic common across all the planners is the Euclidean distance from the goal and the inadmissible heuristics used for MHA* and A-MHA* include an 8-connected Dijkstra search assuming the robot to have zero size and two other progressive heuristics obtained by running 8-connected Dijkstra search on a map created by blocking the narrow passages (passage width $\leq$ robot size).

Metric	A-MHA*		ARA*		ANA*		MHA*	
Size	48	63	48	63	48	63	48	63
SR	100	88	75	70	75	44	100	88
T_i	17	9	15	18	42	23	17	9
T_f	42	39	31	29	111	41	17	9
ϵ_i	25	25	25	25	2e7	3e7	25	25
ϵ_f	7.3	4	8.7	5.1	7.9	3.3	25	25

Table 1: Average statistics for 50 instances of 48 and 63 tile sliding puzzle: SR - Success Rate; T_i - Time to produce the first solution; T_f - Time to produce the final solution; ϵ_i - Reported Initial suboptimality bound; ϵ_f - Reported final suboptimality bound.

Sliding Tiles Puzzle: A widely used consistent heuristic for this domain is the sum of the Manhattan Distance (MD) and Linear Conflict (LC). Similar to the original MHA* paper, the inconsistent heuristics are a weighted sum of the number of misplaced tiles (MT), MD and LC, where the weights are randomly generated during execution.

Experimental results presented in Figure 1 and Table 1 strongly favor A-MHA*. This simple algorithm brings together the benefits of both ARA* and MHA*, thereby maximizing the performance in terms of both solution quality and run-time.

References

Aine, S.; Swaminathan, S.; Narayanan, V.; Hwang, V.; and Likhachev, M. 2016. Multi-heuristic a. *The International Journal of Robotics Research* 35(1-3):224–243.

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Likhachev, M.; Gordon, G. J.; and Thrun, S. 2004. Ara*: Anytime a* with provable bounds on sub-optimality. In *Advances in neural information processing systems*, 767–774.

Pohl, I. 1970. Heuristic search viewed as path finding in a graph. *Artificial intelligence* 1(3-4):193–204.

Van Den Berg, J.; Shah, R.; Huang, A.; and Goldberg, K. 2011. Anytime nonparametric a. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*.