**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 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 $s \in S$ denote the finite set of discrete states over which we search for a path from $s_{\text{start}}$ to $s_{\text{goal}}$. The search typically proceeds by expanding states to generate successors $s' \in \text{Succ}(s)$ 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 $h_i(s), i = 1, \ldots, N$. Let

---

Copyright © 2019, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

---

*These authors contributed equally.

---

We include MHA*, ARA* and Anytime Nonparametric A* (ANA*) (Van Den Berg et al. 2011) in our comparisons.
Algorithm 1 Anytime Multi Heuristic A* algorithm

1: procedure KEY(s, i)
2: return \( g(s) + w_1 \ast h_i(s) \);
3: procedure EXPAND(s, i)
4: Remove s from OPEN\(_i\) ∀ i = 0, 1…N
5: for each \( s' \) in Succ(s)
6: if \( g(s') > g(s) + c(s, s') \)
7: \( g(s') = g(s) + c(s, s') \)
8: if \( s' \) in CLOSED\(_{anch} \)
9: Add \( s' \) to \( I N C O N S \)
10: else
11: Insert/Update \( s' \) in OPEN\(_0\) with KEY(\( s', 0 \))
12: if \( s' \) not in CLOSED\(_{inad} \)
13: for \( i = 1 \) to \( n \)
14: if KEY(\( s', i \)) ≤ w\(_2\) * KEY(\( s', 0 \))
15: Insert/Update \( s' \) in OPEN\(_i\) with KEY(\( s', i \))
16: procedure IMPROVEPATH()
17: while \( f(s_{goal}) > w_2 \ast OPEN_0.Min() \)
18: for \( i = 1 \ldots N \)
19: if(OPEN\(_i\).Min() ≤ w\(_2\) \ast OPEN\(_0\).Min())
20: \( s = OPEN_i.Top() \)
21: EXPAND(s, i) and Insert s in CLOSED\(_{inad} \)
22: else
23: \( s = OPEN_0.Top() \)
24: EXPAND(s, 0) and Insert s in CLOSED\(_{anch} \)
25: procedure MAIN()
26: \( w_1 = \frac{w_2}{2}; w_2 = w_2^2; g(s_{start}) = 0; g(s_{goal}) = \infty; \)
27: for \( i = 0 \ldots N \)
28: OPEN\(_i\) = NULL
29: Insert \( s_{start} \) in OPEN\(_i\) with KEY(\( s, i \))
30: while \( w_1 \geq 1 \) and \( w_2 \geq 1 \)
31: CLOSED\(_{anch} \) = CLOSED\(_{inad} \) = NULL
32: INCONS = NULL
33: IMPROVEPATH()
34: Publish current \( w_1 \ast w_2 \) suboptimal solution
35: if \( w_1 \) = 1 and \( w_2 \) = 1
36: return
37: \( w_i = \max(w_1 - \Delta w_i, 1); i = 1, 2 \)
38: Move states from INCONS into OPEN\(_0\)
39: Copy all states from OPEN\(_0\) to OPEN\(_i\)
40: Update the priorities ∀ \( s \in OPEN_i; \forall i = 0..N \)

3D Path Planning: We plan for a polygonal robot with

<table>
<thead>
<tr>
<th>Metric</th>
<th>A-MHA*</th>
<th>ARA*</th>
<th>ANA*</th>
<th>MHA*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>48</td>
<td>48</td>
<td>48</td>
<td>48</td>
</tr>
<tr>
<td>SR</td>
<td>100</td>
<td>88</td>
<td>75</td>
<td>75</td>
</tr>
<tr>
<td>( T_i )</td>
<td>17</td>
<td>9</td>
<td>15</td>
<td>18</td>
</tr>
<tr>
<td>( T_f )</td>
<td>42</td>
<td>39</td>
<td>31</td>
<td>29</td>
</tr>
<tr>
<td>( \epsilon_i )</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>( \epsilon_f )</td>
<td>7.3</td>
<td>4</td>
<td>8.7</td>
<td>5.1</td>
</tr>
</tbody>
</table>

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; \( \epsilon_i \) - Reported Initial suboptimality bound; \( \epsilon_f \) - Reported final sub-optimality bound.