Multi-Heuristic A*

Sandip Aine
Indraprastha Institute of Information and Technology, New Delhi

Siddharth Swaminathan
Carnegie Mellon University, Pittsburgh

Venkatraman Narayanan
Carnegie Mellon University, Pittsburgh

Victor Hwang
Carnegie Mellon University, Pittsburgh

Maxim Likhachev
Carnegie Mellon University, Pittsburgh

Abstract

We present a novel heuristic search framework, called Multi-Heuristic A* (MHA*), that simultaneously uses *multiple, arbitrarily inadmissible* heuristic functions and one consistent heuristic to search for complete and bounded suboptimal solutions. This simplifies the design of heuristics and enables the search to effectively combine the guiding powers of different heuristic functions. We support these claims with experimental results on full-body manipulation for PR2 robots.

The performance of heuristic search based planners depends heavily on the accuracy of the heuristic function and can degrade severely in the presence of heuristic depression regions, i.e., regions in the search space where the heuristic values do not correlate well with the actual cost-to-goal. Designing a single accurate heuristic function that is also admissible and consistent, is difficult for complex planning problems. In contrast, for many domains, one can easily compute different inadmissible heuristics, each of which can provide complementary guiding power. When used in isolation, these additional heuristics provide little value, as they can neither guarantee completeness, nor can they guarantee efficient convergence. However, a search can consider multiple such heuristics to explore different parts of the search space while using a consistent heuristic to ensure completeness. This may result in faster convergence, if any of these heuristics (or a combination) can guide the search around the depression regions. In this work, we propose an algorithmic framework, called Multi-Heuristic A* (MHA*), that runs multiple searches with different inadmissible heuristics in a manner that preserves guarantees on the suboptimality bounds. We discuss two variants of MHA*, namely, Independent Multi-Heuristic A* (IMHA*) which uses independent g and h values for each search, and Shared Multi-Heuristic A* (SMHA*) which uses different h values but a single g value for all the searches. Related works (Röger and Helmert 2010) have explored the applicability of using multiple heuristics in search. However, unlike our approach, they do not provide any guarantees on completeness/suboptimality. For more details, please refer to the full paper (Aine et al. 2014).

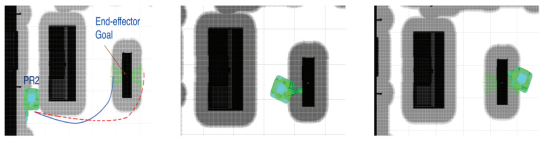
Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

This research was sponsored by the ONR DR-IRIS MURI grant N00014-09-1-1052 and DARPA Computer Science Study Group (CSSG) grant D11AP00275.

Multi-Heuristic A* (MHA*)

We assume that we have n heuristics denoted by h_i for $i = 1..n$, these heuristics can be arbitrarily inadmissible. We also require access to a consistent heuristic (h_0). MHA*s use separate priority queues for each search ($n+1$ queues), denoted by $OPEN_i$, for $i = 0..n$. We use the term *anchor* search to refer to the search that uses h_0, other searches are referred to as *inadmissible* searches.

In the IMHA* algorithm (Figure 2), different heuristics are explored *independently* by running $n+1$ searches. The suboptimality bound is controlled using two variables, w_1 to



(a) Heuristics (b) Admissible Search (c) Inadmissible Search

Figure 1: A full-body manipulation (with PR2) example showing the utility of multiple heuristics. 1a shows two heuristic-guided paths, blue (solid) for an admissible heuristic and red (dotted) for an inadmissible heuristic. The admissible heuristic guides the search to a local minimum (1b), whereas the inadmissible heuristic guides the search along a feasible path that allows the planner to converge quickly (1c).

```
1  procedure Expand(s, i)
2  Remove s from OPEN_i;
3  for each s' in Succ(s)
4    if s' was never visited in the i^th search g_i(s') = ∞; bp_i(s') = null;
5    if g_i(s') > g_i(s) + c(s, s')
6      g_i(s') = g_i(s) + c(s, s'); bp_i(s') = s;
7      if s' has not been expanded in the i^th search
8        insert/update s' in OPEN_i with g_i(s) + w_1 * h_i(s) as priority;
9  procedure IMHA*()
10 for i = 0 to n
11   g_i(s_goal) = ∞; bp_i(s_start) = bp_i(s_goal) = null;
12   g_i(s_start) = 0; OPEN_i = ∅; insert s_start into OPEN_i;
13 while OPEN_0 not empty
14   for i = 1 to n
15     if OPEN_i.Minkey() ≤ w_2 * OPEN_0.Minkey()
16       if g_i(s_goal) ≤ OPEN_i.Minkey() return path pointed by bp_i(s_goal);
17       s = OPEN_i.Top(); Expand(s, i);
18     else
19       if g_0(s_goal) ≤ OPEN_0.Minkey() return path pointed by bp_0(s_goal);
20       s = OPEN_0.Top(); Expand(s, 0);
```

Figure 2: Independent Multi-Heuristic A* (IMHA*)

```
1  procedure Expand(s)
2  Remove s from OPEN_i ∀i = 0..n;
3  for each s' in Succ(s)
4   if s' was never visited  g(s') = ∞; bp(s') = null;
5   if g(s') > g(s) + c(s, s')
6    g(s') = g(s) + c(s, s'); bp(s') = s;
7    if s' has not been expanded in the anchor search
8     insert/update s' in OPEN_0 with g(s) + w_1 * h_0(s) as priority;
9     if s' has not been expanded in any inadmissible search
10      insert/update s' in OPEN_i with g(s) + w_1 * h_i(s) as priority, ∀i = 1..n;
11 procedure SMHA*()
12 g(s_goal) = ∞; bp(s_start) = bp(s_goal) = null; g(s_start) = 0;
13 for i = 0 to n  OPEN_i = ∅; insert s_start into OPEN_i;
14 while OPEN_0 not empty
15  for i = 1 to n
16   if OPEN_i.Minkey() ≤ w_2 * OPEN_0.Minkey()
17    if g(s_goal) ≤ OPEN_i.Minkey()  return path pointed by bp(s_goal);
18    s = OPEN_i.Top(); Expand(s);
19   else
20    if g(s_goal) ≤ OPEN_0.Minkey()  return path pointed by bp(s_goal);
21    s = OPEN_0.Top(); Expand(s);
```

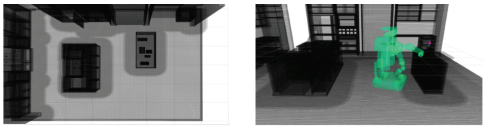
Figure 3: Shared Multi-Heuristic A* (SMHA*)

inflate the heuristics (for all the searches) and w_2 factor to prioritize the inadmissible searches over the *anchor* search. IMHA* runs the inadmissible searches in a round robin manner as long as they explore solutions within w_2 factor of the minimum key of the anchor search. If an inadmissible cannot guarantee the bound, it is suspended, and the *anchor* search is run instead. IMHA* satisfies the following two properties. First, upon termination, the solution cost obtained is bounded by $w_1 * w_2$ factor and second, no state is expanded more than $n + 1$ times.

The primary difference between SMHA* and IMHA* is that in SMHA* (Figure 3) the current path to a given state is shared among all the searches. SMHA* uses a single g value for each state and its expansion procedure updates all the queues simultaneously. SMHA* guarantees the same solution quality bound as IMHA* ($w_1 * w_2$), and achieves this bound with at most 2 expansions per state (IMHA* may expand a state $n + 1$ times). Also, since SMHA* shares the paths among searches, it can use a combination of partial paths to exit from depression regions, which is not possible in IMHA*.

Experimental Results

One of our experiments was in the domain of 12D mobile manipulation planning for the PR2 robot in a kitchen like environment (Figure 4). The planner was provided an initial configuration of the robot as the start state and a goal state containing the 6 DOF position of the final end effector configuration. We computed the consistent heuristic by taking the maximum value between the base heuristic (2D distance from a chosen base circle around the goal) and the end-effector heuristic (3D distance for the end effector configuration). For IMHA*, we computed 2 additional heuristics in the following way. We randomly selected 2 points (with valid IK solutions) on the base circle and ran 2D Dijkstra searches from these 2 points to compute the base distances. We also computed an orientation distance by obtaining the Euclidean distance between the current orientation and the desired orientation, which was to make the robot face the end-effector goal. These inadmissible distances (base and orientation) were then added to the end-effector distance to compute the final heuristic values. For SMHA*, we augment this set by using the base (2D Dijkstra + orientation) and the end-effector heuristics (3D Dijkstra) as two additional heuristics.



(a) (b)

Figure 4: Example of kitchen environments used for our experiments.

	WA*	MHGBFS	MPWA*	EES	IMHA*	SMHA*
SR	31%	76%	36%	27%	70%	81%
SE	1.08	0.78	3.84	1.54	1.58	1.0
RT	0.99	0.91	2.82	1.54	1.41	1.0
SC	0.95	1.57	0.97	0.93	1.09	1.0

Table 1: Comparison between WA*, MHGBFS, MPWA*, EES and MHA*s ($w = 50$ and time limit $= 60secs$). The first row (SR) shows the percentage of instances solved by each planner. The other rows include the results as a ratio between the algorithm in the column heading and SMHA*. Legend: SR - success rate, SE - state expansion ratio, RT - runtime ratio, SC - solution cost ratio.

	PRM	RRT-Connect	RRT*(First)	RRT*(Final)	IMHA*	SMHA*
SR	74%	98%	100%	100%	70%	81%
RT	2.07	0.18	5.39	8.48	1.41	1.00
BD	1.93	1.88	1.36	1.34	1.02	1.00
ED	1.87	1.68	1.27	1.24	0.99	1.00

Table 2: Comparison between MHA*s and sampling based planners (time limit = 60 secs). Legend: SR - Success Rate, RT - runtime ratio, BD - base distance ratio, ED - end effector distance ratio.

Table 1 shows the results comparing WA*, EES (Thayer and Ruml 2011), MPWA* (Valenzano et al. 2010), MHGBFS (Röger and Helmert 2010) with the MHA*s. The results clearly show that MHA*s and MHGBFS perform much better than WA*/MPWA*/EES for this domain, highlighting the efficacy of using multiple heuristics since a single heuristic function often suffers from local minima due to robot's orientation, position of the other objects, etc. MHGBFS performs comparably with SMHA* in terms of instances solved, however it is worse in terms of solution costs.

Table 2 depicts the results comparing MHA*s with 3 sampling based algorithms, PRM, RRT-Connect and RRT* (Şucan, Moll, and Kavraki 2012). The results show that MHA*s perform reasonably well when compared to the sampling based planners, they are comparable in terms of runtime and better in terms of solution quality.

References

Aine, S.; Swaminathan, S.; Narayanan, V.; Hwang, V.; and Likhachev, M. 2014. Multi-Heuristic A*. In *Proceedings of the Robotics: Science and Systems (RSS)*.

Röger, G., and Helmert, M. 2010. The more, the merrier: Combining heuristic estimators for satisficing planning. In *ICAPS*, 246–249.

Şucan, I. A.; Moll, M.; and Kavraki, L. E. 2012. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine* 19(4):72–82.

Thayer, J. T., and Ruml, W. 2011. Bounded suboptimal search: A direct approach using inadmissible estimates. In *IJCAI*, 674–679.

Valenzano, R. A.; Sturtevant, N. R.; Schaeffer, J.; Buro, K.; and Kishimoto, A. 2010. Simultaneously searching with multiple settings: An alternative to parameter tuning for suboptimal single-agent search algorithms. In *ICAPS*, 177–184.