Multi-Resolution A*

Wei Du, Fahad Islam, and **Maxim Likhachev**
The Robotics Institute
Carnegie Mellon University
{wdu2, fi, maxim}@cs.cmu.edu

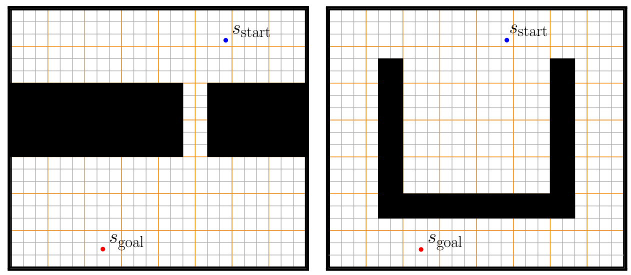
Abstract

Heuristic search-based planning techniques are commonly used for motion planning on discretized spaces. The performance of these algorithms is heavily affected by the resolution at which the search space is discretized. Typically a fixed resolution is chosen for a given domain. While a finer resolution allows better maneuverability, it significantly increases the size of the state space, and hence demands more search efforts. On the contrary, a coarser resolution gives a fast exploratory behavior but compromises on maneuverability and the completeness of the search. To effectively leverage the advantages of both high and low resolution discretizations, we propose Multi-Resolution A* (MRA*) algorithm, that runs multiple weighted-A*(WA*) searches having different resolution levels simultaneously and combines the strengths of all of them. In addition to these searches, MRA* uses one anchor search to control the inadmissible expansions from these searches. We show that MRA* is resolution complete and bounded suboptimal with respect to the anchor resolution search space. We performed experiments on several motion planning domains including 2D, 3D grid planning and 7 DOF manipulation planning and compared our approach with several search-based and sampling-based baselines.

1 Introduction

Search-based planners are known to be sensitive to the size of state spaces. The three main factors that determine the size of a state space are the state dimension, the resolution at which each dimension is discretized and the size of the environment or the map (Elbanhawi and Simic 2014). The size of state spaces grow exponentially with increased dimension and polynomially with increased resolution.

As an example, consider a large sized map most of which is free space, yet it has a number of narrow passages, which the planner has to find paths through. Fig. 1 shows two snippets from this map discretized at two resolution levels. For the example snippet shown in Fig. 1(a), to find a path from s_{start} to s_{goal}, a search with the coarse resolution space will fail since the narrow passage enforces the requirement of having a high resolution for completeness sake. Not only does a low resolution space weakens the completeness guarantee, but also sacrifices solution quality.



(a) Narrow passage (b) Cul-de-sac

Figure 1: Discretization of maps with high (grey) and low (orange) resolutions. To the left, solution via transitions only through coarse cells does not exist. To the right, however, it is cumbersome for planner to escape local minimum on a high resolution map.

Consider another region of this example map shown in Fig. 1(b). For this problem instance, it is evident that the high resolution search would require a lot more expansions before it escapes the local minimum than the lower resolution search. Our key insight is that, settling on a single resolution discretization trades off search efficiency with completeness and optimality guarantees. To this end, we propose the Multi-Resolution A* (that we shorten as MRA*) algorithm to combine the advantages of different resolution discretizations by employing multiple weighted-A* (WA*) (Pohl 1973) searches that run on the different resolution state spaces simultaneously.

MRA* uses multiple priority queues that correspond to searches at each resolution level and shares information between them. States from different discretization that coincide are considered as the same state and thus shared between corresponding queues. Our approach bears some resemblance to Multi-Heuristic A* (MHA*) algorithm (Aine et al. 2016); MHA* uses multiple possibly inadmissible heuristics in addition to a single consistent anchor heuristic which is used to provide suboptimality bounds. Instead of taking advantage of multiple heuristics in different searches, we leverage multiple state spaces at different resolutions. To provide suboptimality guarantees we use an anchor search which runs on a particular resolution space. We prove that our algorithm is resolution complete (LaValle 2006) in the anchor resolution space and bounded suboptimal with respect to the optimal path in the anchor resolution space.

We conduct experiments on 2D, 3D environments and for a 7-DOF single-arm manipulation problems, and compare MRA* with other search-based algorithms and sampling-based algorithms. The results verify that MRA* outperforms other algorithms for various performance metrics.

2 Related Work

Motion planning in high dimensional and large-scale domains is challenging both for search-based and sampling-based approaches (Petrovic 2018).

Sampling-based methods are popular candidate for high-dimensional motion planning problems. They have an advantage that they do not rely on a resolution, rather they use random sampling to discretize the state space. Randomized methods such as RRT (LaValle 2006) and RRT-Connect (Jr. and LaValle 2000) quickly explore high-dimensional space due to their random sampling feature. Although fast, these algorithms are non-deterministic and provides no guarantees on the quality of solutions that they found. Optimal variants such as RRT* (Karaman and Frazzoli 2011) provide asymptotic optimality guarantees, namely, they reach optimal solution as the number of samples grows to infinity. Following RRT*, a family of algorithms including FMT* (Janson et al. 2015), RRT*-Smart (Islam et al. 2012) and Informed-RRT* (Gammell, Srinivasa, and Barfoot 2014) were developed to improve the convergence rate of RRT*. These algorithm improve the quality of the solutions over time but do not provide bounds on the solution quality. Moreover they also give inconsistent solutions due to their inherent randomised behavior.

It is well-known that search-based planners suffer from the *curse of dimensionality* (Bellman 1957). They rely on a specific space discretization, the choice of which largely affects the computational complexity and properties of the algorithm. Several methods have been proposed to alleviate this problem on discrete grids. Moore et al. came up with the Parti-game algorithm (Moore and Atkeson 1995), which adaptively discretizes the map with high resolution at the border between obstacles and free space and low resolution on large free space. Similarly, this notion is implemented via quad-tree search algorithms (Garcia, Kapadia, and Badler 2014; Yahja et al. 1998). These algorithms are memory efficient in sparse environments, however, in cluttered environments, these approaches show little to no advantages over uniformly discretized map because of the overhead in book-keeping of the graph edges. Furthermore, as these methods require searching on an explicit graph, they are not scalable to high dimensional planning problems because of the memory requirements.

In addition to grid search, search over implicit graphs formulated by state lattices (Pivtoraiko and Kelly 2005) is ubiquitous in both navigation and planning for manipulation (Cohen, Chitta, and Likhachev 2010). These methods rely on motion primitives which are short kinematically feasible motions that the robot can execute. In (Likhachev and Ferguson 2009), graph search for autonomous vehicles was run on a multi-resolution lattice state space. More specifically, they used high resolution space close to the robot or goal region and a low resolution action space elsewhere. Similarly, the Hierarchical Path-Finding A* (HPA*) algorithm (Botea, Müller, and Schaeffer 2004) pre-processes maps into different levels of abstractions. Then the full solution is constructed by concatenating segments of trajectories within a local cluster which belongs to higher level abstraction path. This approach relies on the condition that there is a smooth transition between high and low resolution abstractions. Besides, these hierarchical structures require large memory footprint for maintaining the different abstractions and have significant computational overhead for pre-processing. On the contrary, MRA* runs search over an implicitly constructed graph (generated on the fly during search) and therefore, it requires less memory and no pre-computation overhead.

Another class of methods plan in non-uniform state dimension and action to reduce the size of search state spaces (Cohen et al. 2011; Cohen, Chitta, and Likhachev 2014). Cohen et al. observed that not all the joints of a manipulator need to be active throughout the search, for example the joints at the end-effector might only be required to move near the goal region. By restricting the search dimension in this manner, they gain considerable speedups. Though efficient, this approach could potentially sabotage the completeness of the search. To overcome this limitation, planning with adaptive dimensionality (Kalin Gochev and Likhachev 2013; Vemula, Mülling, and Oh 2016) allows searching in lower dimension most of the time and only requires searching in the high dimension when necessary. On related lines, (Brock and Kavraki 2001) decomposes the original problem into several high-dimensional and low-dimensional sub-problems in a divide-and-conquer fashion. Our approach is different from these methods, since we decompose the search space into multiple resolutions instead of multiple dimensions.

3 Multi-Resolution A*

MRA* employs multiple WA* searches in different resolution spaces (high and low) simultaneously and shares the states that coincide on the respective discretizations. To gain more benefit out of the algorithm, the resolutions should be selected such that more sharing is facilitated. If no sharing is allowed at all, the algorithm would degenerate into several independent searches and the solution will be returned by any search that would satisfy the termination criterion first. In addition to these searches, MRA* uses an anchor search which is an optimal A* search, to provide bounds on the solution quality. In the remainder of this section we formally describe our algorithm. We will also propose a variant of the algorithm which differs in the representation of the anchor search. We will also discuss the theoretical properties of both the algorithms.

3.1 Problem Definition and Notations

In the following S denotes a discretized domain. Given a start state s_{start} and a goal state s_{goal} , the planning problem is defined as finding a collision free path from s_{start}

Algorithm 1 Multi-Resolution A*

1: **procedure** MAIN
2: $g(s_{\text{start}}) = 0$; $g(s_{\text{goal}}) = \infty$
3: $bp(s_{\text{start}}) = bp(s_{\text{goal}}) =$ **null**
4: **for** $i = 0, ..., n$ **do**
5: OPEN$_i \longleftarrow \emptyset$
6: CLOSED$_i \longleftarrow \emptyset$
7: **if** $i \in$ GETSPACEINDICES(s_{start}) **then**
8: Insert s_{start} in OPEN$_i$ with KEY(s, i)
9: **while** OPEN$_0 \neq \emptyset$ **do**
10: $i \longleftarrow$ CHOOSEQUEUE( )
11: **if** OPEN$_i$.MINKEY() $\leq \omega$ * OPEN$_0$.MINKEY() **then**
12: **if** $g(s_{\text{goal}}) \leq OPEN_i$.MINKEY() **then**
13: Return path pointed by $bp(g(s_{\text{goal}}))$
14: **else**
15: s = OPEN$_i$.Pop()
16: EXPANDSTATE(s, i)
17: Insert s into CLOSED$_i$
18: **else**
19: **if** $g(s_{\text{goal}}) \leq OPEN_0$.MINKEY() **then**
20: Return path pointed by $bp(g(s_{\text{goal}}))$
21: **else**
22: s = OPEN$_0$.Pop()
23: EXPANDSTATE$(s, 0)$
24: Insert s into CLOSED$_0$

Algorithm 2 ExpandState

1: **procedure** KEY(s, i)
2: **if** $i = 0$ **then**
3: **return** $g(s) + h(s)$
4: **else**
5: **return** $g(s) + \varepsilon h(s)$
6: **procedure** EXPANDSTATE(s, i)
7: **for all** $s' \in$ SUCCS(s, i) **do**
8: **if** s' was never generated **then**
9: $g(s') = \infty$; $bp(s') =$ **null**;
10: **if** $g(s') > g(s) + c(s, s')$ **then**
11: $g(s') = g(s) + c(s, s')$; $bp(s') = s$
12: **if** $s' \notin$ CLOSED$_0$ **then**
13: **if** $0 \in$ GETSPACEINDICES(s') **then**
14: Insert/Update s' in OPEN$_0$ with KEY$(s', 0)$
15: **for** each $i \in$ GETSPACEINDICES(s') - $\{0\}$ **do**
16: **if** $s' \notin$ CLOSED$_{\text{i}}$ **then**
17: Insert/Update s' in OPEN$_i$ with KEY(s', i)

and s_{goal} in S. The cost from s_{start} to a state s is denoted as $g(s)$, optimal cost to come is denoted by $g^*(s)$ and $bp(s)$ is a back-pointer which points to the best predecessor of s (if one exists). The function $c(s, s')$ denotes non-negative edge cost between any pair of states in S. Throughout the algorithm the anchor search and its associated data structures are indexed by 0 whereas other searches are denoted with indices 1 through n.

We have multiple action sets corresponding to different resolution spaces. We also define a full action space which is the union of all action sets:

$$A_{\text{full}} = A_0 \cup A_1 \cup \ldots \cup A_n.$$

where A_i is a set of actions for resolution i. SUCCS(s, i) returns all successors of s for resolution i. GETSPACEINDICES(s) returns a list of indices of all the spaces which the state s coincides with. Furthermore, we assume that we have access to a consistent heuristic function $h(s)$. Each WA* search uses a priority queue OPEN$_i$ with the priority function KEY$_i(s)$ and a list of expanded states CLOSED$_i$. Additionally, each queue has a function OPEN$_i$.MINKEY() which returns the minimum KEY value for the ith queue.

3.2 Algorithm

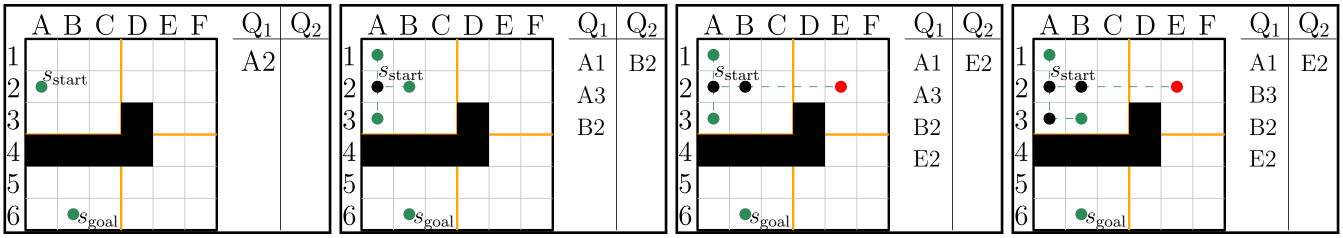
The main algorithm is presented in Alg. 1. The lines 2- 8 initialize the g values and back pointers of s_{start} and s_{goal}, and OPEN and CLOSED for each queue and insert s_{start} into all queues with which s_{start} coincides with the corresponding priority values.

The algorithm runs until OPEN$_0$ gets empty (line 9) or any of the two termination criteria (lines 12 or 19) are met. In line 10, we employ a scheduling policy to make decision on from which queue to expand a state in current iteration. This scheduling policy could be a round-robin strategy, Dynamic Thompson Sampling (DTS) policy or other scheduling policies, as is suggested in (Phillips et al. 2015) [1]. The condition in line 11 controls the inadmissible expansions from other queues by expanding from the anchor whenever the condition fails. The expansions from the anchor queue monotonically increase OPEN$_0$.MINKEY$()(s)$ as the anchor is a pure A* search, allowing more states to be expanded from the other queues. The minimum priority state is popped from OPEN$_i$ and then added to the corresponding CLOSED$_i$.

Details of a state expansion are presented in Alg. 2. The EXPANDSTATE(s, i) function "partially" expands state s in the search i by using actions A_i. If the successors of s are duplicates of states in other spaces, they are inserted or updated in the corresponding searches as well. This is how the paths or the g values of the states are shared between the different searches. In this procedure, the condition at Line 10 indicates that a state will only be updated in a queue if its g value is improved. If a state is closed for anchor search it will not be inserted/updated in any queue. If a state is not closed for the anchor, it can be inserted/updated in other queues if it is not closed for those queues (lines 12- 16). If a state is closed in a non-anchor queue, it could be reinserted and therefore re-expanded by the anchor. Note that a state is only inserted in a queue if it coincides with its discretization (see lines 13 and 15).

Fig. 2 provides a simple 2D illustration of the MRA* algorithm. We use two resolutions (high and low) in this example and MRA* alternatively expands states from the two queues. The cell size (the side length of one cell) of the low resolution space is 3 times the size of the high resolution space. For the sake of simplicity, we assume that the suboptimality bound ω is very high such that anchor queue is never

[1] In DTS policy, the selection of a queue is viewed as a *multi-arm bandit* problem (Gupta, Granmo, and Agrawala 2011), where the reward from a "bandit" is equal to the search progress made by the decision, reflected in the decrease of chosen queue's top state's heuristic value.

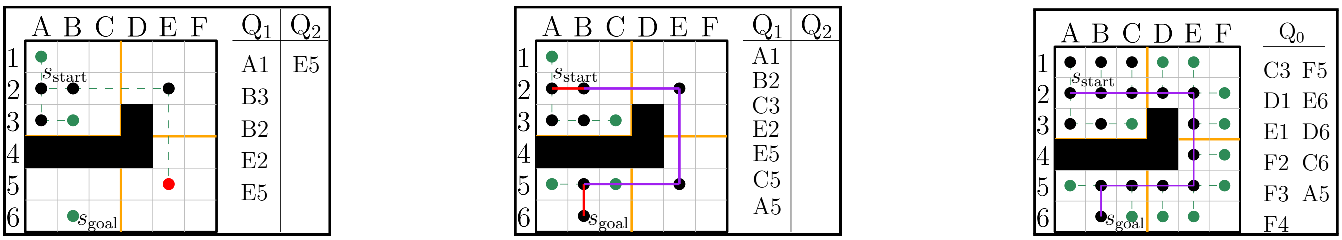


(a) MRA* is initialized. State s_{start} (A2) is inserted into Q_1.

(b) State A2 is expanded by high resolution search. Since state B2 lies at the center of a coarse cell, it is also inserted into Q_2.

(c) State B2 is expanded by low resolution search. The successor E2 is inserted into both queues.

(d) State A3 is expanded by high resolution search.



(e) State E2 is expanded by low resolution search and the successor E5 is inserted into both queues.

(f) The last step when s_{goal} is expanded by high resolution search. A solution (solid line segments) is found with 8 expansions in total.

(g) The final status of a high resolution search. The same solution (purple) is found with 17 expansions.

Figure 2: Illustration of MRA* algorithm—Thick (orange) lines and thin (grey) lines show the low and high resolution grids respectively. MRA* initializes in Fig. 2(a). Figs. 2(b) to 2(e) show the first four expansions of MRA* and Fig. 2(f) shows the last expansion when the search terminates. OPEN lists for the high and low resolution searches are denoted as Q_1 and Q_2 respectively. Expanded states are shown in black, states in OPEN lists are shown in green and the states that coincide between the two spaces are shown in red. The path returned by MRA* is composed of edges from both high (red) and low (purple) resolution spaces. Fig. 2(g) illustrates the behaviour of WA* search only in the high resolution grid.

expanded i.e the condition in line 11 is never violated. We also assume that the weights ε are high enough that the WA* searches are purely greedy. Fig. 2(g) shows the result if we would only run a single high-resolution search for the same example for comparison. It is evident that benefited from the sharing feature between multiple resolution spaces, MRA* found the solution with much less expansions than the high resolution WA* search.

3.3 Analysis

Theorem 1. *During the execution of MRA*, a) a state is never re-expanded more than once in the same resolution space and b) a state expanded in anchor is never re-expanded in the same resolution space.*

If s is expanded by the search i it is inserted in CLOSED_i (Alg. 1, Line 17) and Alg. 2, Line 16 ensures that s is never reinserted in OPEN_i, hence it will never be re-expanded by search i. However, s can be inserted in OPEN_0 if it is not closed in the anchor search. Once expanded by the anchor, Line 12 will not allow its re-expansion by any search.

Theorem 2. *MRA* is resolution-complete in the anchor resolution space.* [2]

[2] An algorithm is considered complete for a resolution if it is guaranteed to return a solution if one exists in finite time on the state space discretized by that resolution.

This holds true because the algorithm terminates until it finds a solution or the anchor state space is exhausted (Alg. 1, line 9)

Theorem 3. *In MRA*, solution returned by any search i with total cost $g_i(s_{\text{goal}})$ is bounded as:*

$$g_i(s_{\text{goal}}) \leq \omega * g_0^*(s_{\text{goal}})$$

where $g_0^*(s_{\text{goal}})$ is the optimal solution with respect to anchor resolution.

Proof. If the anchor search terminates at Alg. 1, line 20 then because the anchor search is an optimal A* search, from (Pearl 1984), we have

$$\begin{aligned} g_0(s_{\text{goal}}) &= g_0^*(s_{\text{goal}}) \\ &\leq \omega * g_0^*(s_{\text{goal}}) \end{aligned} \tag{1}$$

If any other search terminates (Alg. 1, line 13), then from lines 11 and 12, and because the anchor search is A* search we have,

$$\begin{aligned} g_i(s_{\text{goal}}) &\leq \omega * \text{OPEN}_0.\textsc{MinKey}() \\ &\leq \omega * g_0^*(s_{\text{goal}}) \text{ From (Pearl 1984)} \end{aligned} \tag{2}$$

□

3.4 Variant of MRA*

In this section we describe another variant of MRA* and discuss its associated properties. Instead of having a single resolution space for the anchor search, we could construct a space which is the union of all other resolution spaces. In other words, this anchor graph would be a super-set graph of all other graphs. This variant introduces the following changes to MRA*'s properties:

- The variant provides stronger completeness guarantee, that is, the algorithm would be complete with respect to the union space of all the other queues and not a single resolution space.
- The variant provides suboptimality bounds on the union space. As a result, the quality of the solution found by the algorithm will be bounded by the optimal solution in the union space and not in one resolution space.
- For this variant, a state is never fully expanded at most twice; A "full expansion" means expanding one state s with the action set A_{full} . This holds because a state can be partially expanded by any non-anchor queue and then fully re-expanded by the anchor queue. However if a state is fully expanded by the anchor, it will never be re-expanded from any queue.

4 Experiments and Results

We evaluate our algorithm on 2D, 3D and 7D domains and report comparisons with different search-based and sampling-based planning approaches in terms of planning time, solution cost, number of expanded states (only for search-based algorithms) and success rates. All experiments were run on an Intel i7-3770 CPU (3.40 GHz) with 16GB RAM. In all experiments, we set a timeout of 120 seconds. For 2D and 3D spaces, we used 8-connected and 26-connected grids. For 7D experiments we used PR2 robot's single-arm and constructed the graph using a manipulation lattice (Cohen et al. 2011). The heuristics used for 2D and 3D domains are octile distance and euclidean distance respectively. For manipulation problems, the heuristic was computed by running a backward 3D Dijkstra's from the end-effector's position at the 6-DoF goal pose. We used Euclidean distance as cost function for 2D and 3D, and Manhattan distance in joint angles for 7D. For all the domains, the anchor search of MRA* is set as the highest resolution space. As the queue selection policy, we used round-robin policy for 2D and 3D, and DTS for the 7D domain. For every domain, we plot statistics showing improvements of MRA* over baselines, where improvements are computed as the average metric values of baselines divided by that of MRA*'s (Fig. 5). For these plots we only report results for common success tests. In addition, we also show tabulated results for all the metrics (Table 1).

4.1 2D Space Planning Results

Domain: We used two different maps discretized into $10,000 \times 10,000$ cells as the highest resolution discretization. Additionally, we have middle and low resolutions whose cells are 7 and 21 times the size of highest resolution cells respectively. The benchmark maps are from *Moving AI Lab* (Sturtevant 2012) *Starcraft* category. For each map, we have 100 randomly generated start and goal pairs. We compare our algorithm with four baselines, three of which search over implicit graph. These are WA* with Multiple Resolutions (WA-MR), WA* with highest resolution (WA-High) and with lowest resolution (WA-Low). WA-MR's action space uses the union of all the resolution spaces in a single queue. The fourth baseline searches over a pre-constructed explicit graph that is the quad-tree search method (Garcia, Kapadia, and Badler 2014) (QDTree). In quad-tree experiments, to book-keep neighbors of a grid, we followed the methods suggested in (Li and Loew 1987b; 1987a). For our algorithm, we set the ε and ω values both to 3.0. For other search-based algorithms, we set the weight to 3.0 as well, which would enforce the same suboptimality bounds for all the algorithms.

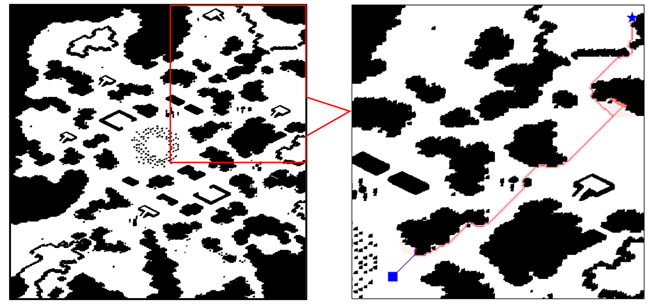


Figure 3: A 2D solution example. The planner is planning from start (square) to goal (star). The red dots are expanded states and the blue line is the solution returned by planner.

Results and Analysis: The results of 2D planning are presented in Fig. 5(a) and Table. 1a. A test map and a sample solution from MRA* is shown in Fig. 3. In the top-right region of the right figure, we can see that MRA* sparsely searched the local minimum region and exited swiftly. This is consistent with the behaviour that we described in Fig. 2.

Our algorithm outperforms WA-MR and WA-High in speed and number of expansions as shown in Fig. 5(a). The speedup comes from the fact that WA-MR performs a full expansion of a every state which is expensive whereas MRA* only uses partial expansions. WA-High searches only in the highest resolution which is also expensive, MRA* on the other hand leverages the low resolution space to quickly escape local minima and uses the high resolution space to plan through narrow passages. WA-Low is faster than MRA* since it only searches in the lowest resolution space, but it also makes it incomplete with respect to the high resolution space. This is verified by the lowest success rate in Table. 1a. QDTree is faster compared to MRA* because the quad-tree map discretization is done in such a way that large open spaces are not further discretized into smaller units, this helps to keep the size of state space small. However the graph construction step is computationally expensive and had an average pre-computation time of 36 seconds for the two maps. The quality of solutions as indicated

Table 1: 2D and 3D planning results.

(a) 2D Planning Results (Map1 & Map2)

	Map1					Map2				
Algorithm	MRA*	WA-MR	WA-High	WA-Low	QDTree	MRA*	WA-MR	WA-High	WA-Low	QDTree
Success Rate (%)	**100**	**100**	**100**	95.5	**100**	**100**	98.99	98.99	94.95	**100**
Mean Time (s)	0.61	5.72	5.62	**0.09**	0.15	4.14	18.23	17.73	**0.22**	0.44
Mean Cost (m)	**324.71**	325.76	326.32	326.71	341.49	**377.91**	379.51	382.35	380.55	396.93

(b) 3D Planning Results (Map1 & Map2)

	Map1				Map2			
Algorithm	MRA*	WA-MR	WA-High	WA-Low	MRA*	WA-MR	WA-High	WA-Low
Success Rate (%)	100	100	100	100	100	100	100	100
Mean Time (s)	2.91	19.01	18.88	**0.06**	4.14	24.16	13.71	**0.07**
Mean Cost (m)	40.20	38.38	**37.04**	40.20	32.41	30.45	**28.83**	31.89

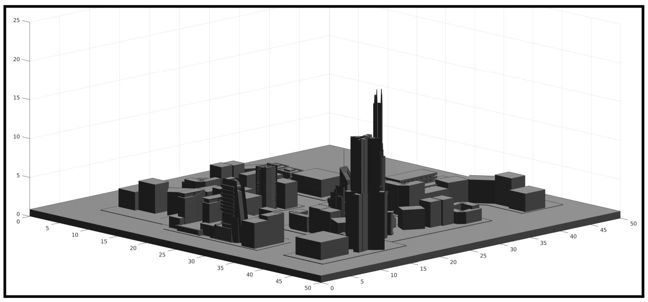


Figure 4: A mesh model of city used as a planning scene for 3D planning.

by the average solution costs in Table. 1a for each algorithm is comparable except QDTree which relatively shows higher costs. This is because QDTree has very coarse discretization in free spaces.

4.2 3D Space Planning Results

Domain: For 3D also we used two maps, one of them is shown in Fig. 4. The other map contains outdoor scenes such as mountains and buildings etc. In the highest resolution, the maps are discretized to a grid of size $1000 \times 1000 \times 400$ cells. Similar to 2D spaces, we have middle and low resolutions that are 9 and 27 times the size of the highest resolution respectively. There are 50 trails in total where start and goal pairs for each trial are randomly assigned. For 3D experiments we only compared with the baselines which search on implicit graphs i.e. WA-MR, WA-High and WA-Low as the overhead of constructing the explicit abstraction for this domain is very high. In our algorithm, we set the ε and ω value both to 3.0. For other search based algorithms, we set the weights to 3.0 as well.

Results and Analysis: The results for scene Fig. 4 are presented in Fig. 5(b). With the same branching factor, WA* in coarse resolution space is significantly faster. As mentioned earlier, the low resolution implementation is incomplete and the suboptimality bounds are also weaker, which results in lower success rate and poor quality solutions. Regarding planning times, MRA* is the fastest as it leverages the different resolution spaces intelligently to quickly find solutions.

For WA-MR, as it performs full state expansions the branching factor becomes very large in 3D i.e. 78, which deteriorates it's performance (see Table. 1b). In terms of solution cost, MRA* generates solutions slightly worse than WA-MR and WA-High, yet still bounded by the same suboptimality bound.

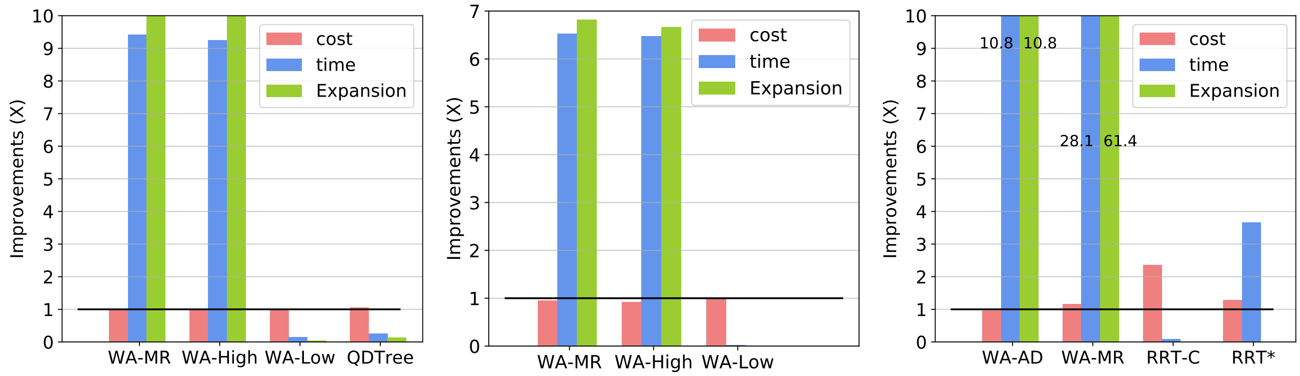
4.3 7D Space Planning Results

For 7D domain implementation we used an adaptation of SMPL[3].

Domain: We used PR2 robot's 7DoF arm for this domain. We ran the experiments on four different benchmark scenarios (Cohen, Chitta, and Likhachev 2014) as in Fig. 6. The start and goal pairs were randomly generated for 70 trails for each scene. We used RRT-Connect (RRT-C) and RRT* as the sampling-based planning baselines. In addition, we tested with WA-MR and WA* with adaptive dimensionality search (Kalin Gochev and Likhachev 2013) (WA-AD) as search-based planning baselines. The implementations of sampling-based approaches are used from Open Motion Planning Library (OMPL) (Şucan, Moll, and Kavraki 2012). For RRT* we report the results for the first solution found. For search-based algorithms, we set the weights for WA* search to be 25. In our algorithm, we set the ε and ω value to 20 and 25 respectively.

Motion Primitives: A base set of 14 motion primitives are provided and categorized into classes with low, middle and high resolutions: M_{low}, M_{middle}, M_{high}. Each motion primitive changes the position of one joint in both directions by an amount corresponding to the resolution. In M_{low}, M_{middle} and M_{high} each action corresponds to a joint angle change of $27°$, $9°$ and $3°$ respectively. In addition to the static motion primitives, adaptive actions are gen-

[3]https://github.com/aurone/smpl

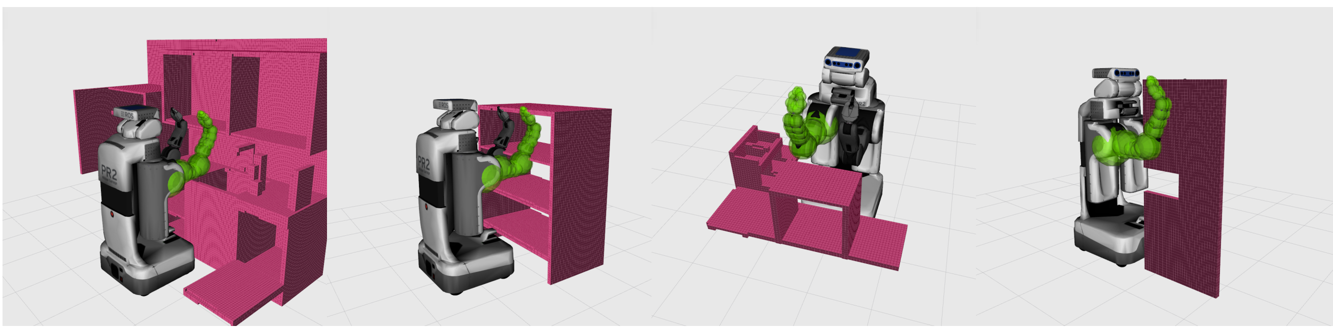


(a) The results of 2D planning.

(b) 3D planning results in scene Fig. 4

(c) 7D planning in scene Industrial.

Figure 5: Improvements of MRA* over baseline algorithms



(a) Kitchen

(b) Bookshelf

(c) Industrial

(d) Narrow Passage

Figure 6: The planning scenes of single-arm manipulation problem.

erated online via inverse kinematics computation (Cohen et al. 2011) to *snap* end-effector to the goal pose when the expanded state is within a small threshold distance to the goal position.

Results and Analysis: We show the experimental results for the *Industrial* scene (Fig. 6(c)) presented in Fig. 5(c). The statistics for the other scenes are very similar and are omitted. In terms of planning times, MRA* outperforms all the baselines except RRT-Connect. MRA* shows over an order of magnitude improvements over WA-AD and WA-MR in planning times and number of expansions, indicating that the performance gains are higher in higher dimension domains. With respect to solution cost, MRA* performs no worse than any other algorithm on common succeeded trials.

From the results documented in Table. 2, MRA* has the highest success rates across all the scenes. Although MRA* is slower than RRT-Connect in terms of solution costs, MRA* (and other search-based baselines) consistently show better solution qualities then RRT-Connect and even RRT*. While WA-MR performs worst in terms of planning time and success rate, it consistently provides the best quality solutions, which could be explained by the fact that WA-MR searches in the graph which is the union of all resolution spaces, and has stricter suboptimality bounds.

5 Conclusion and Future Work

We presented a heuristic search-based algorithm that utilises multiple search spaces implicitly constructed with different resolutions and shares information between them. We show that MRA* is resolution complete on an anchor resolution and the solution returned by MRA* is bounded by the optimal solution in the anchor resolution space. We show that MRA* presents significant performance improvements over the baselines on large 2D, 3D domains and high-dimensional motion planning problems, most importantly in terms of success rates which are consistently high across all the domains and experiments.

While the results are promising, we believe that there is scope for further improvements. Possible future directions can be 1) using multiple heuristics within the different resolutions searches to speed up the search 2) adding dynamic snap motions/primitives for efficient sharing between the different spaces 3) using a large ensemble of resolution spaces and optimizing for the scheduling policy and 4) using the multi-resolution framework for other bounded suboptimal search algorithms such as Optimistic Search (Thayer and Ruml 2008) or search with different priority functions (Chen and Sturtevant 2019).

Table 2: 7D planning results on 4 scenes.

	Kitchen					Bookshelf				
Algorithm	MRA*	WA-AD	WA-MR	RRT-C	RRT*	MRA*	WA-AD	WA-MR	RRT-C	RRT*
Success Rate (%)	**96.83**	49.21	46.031	95.83	75.00	**98.36**	57.38	47.54	89.79	44.00
Mean Time (s)	3.48	12.55	9.19	**0.006**	1.04	1.23	8.44	11.62	**0.13**	9.74
Mean Cost (rad)	7.72	6.22	**5.96**	15.49	8.16	11.30	9.74	**10.38**	28.54	15.70
Processed Mean Cost (rad)	7.23	5.22	**5.26**	8.9	7.25	10.80	**9.13**	9.15	16.93	13.59

	Industrial					Narrow Passage				
Algorithm	MRA*	WA-AD	WA-MR	RRT-C	RRT*	MRA*	WA-AD	WA-MR	RRT-C	RRT*
Success Rate (%)	**96.92**	72.31	15.38	89.83	62.07	**100**	50.00	40.91	96.22	67.27
Mean Time (s)	2.74	7.61	15.48	**0.29**	9.84	3.31	7.98	15.21	**0.05**	4.70
Mean Cost (rad)	13.07	12.77	**11.10**	29.26	16.38	11.68	10.71	**10.12**	20.90	14.39
Processed Mean Cost (rad)	12.33	11.20	**10.53**	16.29	13.77	11.41	10.60	**9.91**	12.42	12.20

References

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

Bellman, R. 1957. *Dynamic Programming*. Princeton University Press.

Botea, A.; Müller, M.; and Schaeffer, J. 2004. Near optimal hierarchical path-finding. *Journal of game development* 1(1):7–28.

Brock, O., and Kavraki, L. E. 2001. Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional spaces. In *IEEE International Conference on Robotics and Automation (ICRA)*, 1469–1474.

Chen, J., and Sturtevant, N. R. 2019. Conditions for avoiding node re-expansions in bounded suboptimal search. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, 1220–1226.

Cohen, B. J.; Subramania, G.; Chitta, S.; and Likhachev, M. 2011. Planning for manipulation with adaptive motion primitives. In *IEEE International Conference on Robotics and Automation (ICRA)*, 5478–5485.

Cohen, B. J.; Chitta, S.; and Likhachev, M. 2010. Search-based planning for manipulation with mootion primitives. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2902–2908.

Cohen, B. J.; Chitta, S.; and Likhachev, M. 2014. Single- and dual-arm motion planning with heuristic search. *The International Journal of Robotics Research (IJRR)* 33(2):305–320.

Elbanhawi, M., and Simic, M. 2014. Sampling-based robot motion planning: A review. *IEEE Access* 2:56–77.

Gammell, J. D.; Srinivasa, S. S.; and Barfoot, T. D. 2014. Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2997–3004.

Garcia, F. M.; Kapadia, M.; and Badler, N. I. 2014. Gpu-based dynamic search on adaptive resolution grids. In *IEEE International Conference on Robotics and Automation (ICRA)*, 1631–1638.

Gupta, N.; Granmo, O.; and Agrawala, A. K. 2011. Thompson sampling for dynamic multi-armed bandits. In *International Conference on Machine Learning and Applications and Workshops (ICMLA)*, 484–489.

Islam, F.; Nasir, J.; Malik, U.; Ayaz, Y.; and Hasan, O. 2012. RRT*-smart: Rapid convergence implementation of RRT* towards optimal solution. In *IEEE International Conference on Mechatronics and Automation*, 1651–1656. IEEE.

Janson, L.; Schmerling, E.; Clark, A. A.; and Pavone, M. 2015. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *The International Journal of Robotics Research (IJRR)* 34(7):883–921.

Jr., J. J. K., and LaValle, S. M. 2000. RRT-Connect: An Efficient Approach to Single-Query Path Planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 995–1001.

Kalin Gochev, A. S., and Likhachev, M. 2013. Incremental planning with adaptive dimensionality. In *International Conference on Automated Planning and Scheduling (ICAPS)*.

Karaman, S., and Frazzoli, E. 2011. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research (IJRR)* 846–894.

LaValle, S. M. 2006. *Planning algorithms*. Cambridge university press.

Li, S., and Loew, M. H. 1987a. Adjacency detection using quadcodes. *Communications of the ACM* 30(7):627–631.

Li, S., and Loew, M. H. 1987b. The quadcode and its arithmetic. *Communications of the ACM* 30(7):621–626.

Likhachev, M., and Ferguson, D. 2009. Planning long dynamically feasible maneuvers for autonomous vehicles. *The International Journal of Robotics Research (IJRR)* 28(8):933–945.

Moore, A. W., and Atkeson, C. G. 1995. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. *Machine Learning* 21(3):199–233.

Pearl, J. 1984. *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Pub. Co., Inc., Reading, MA.

Petrovic, L. 2018. Motion planning in high-dimensional spaces. *Computing Research Repository (CoRR)* abs/1806.07457.

Phillips, M.; Narayanan, V.; Aine, S.; and Likhachev, M. 2015. Efficient search with an ensemble of heuristics. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, 784–791.

Pivtoraiko, M., and Kelly, A. 2005. Generating near minimal spanning control sets for constrained motion planning in discrete state spaces. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3231–3237.

Pohl, I. 1973. The avoidance of (relative) catastrophe, heuristic competence, genuine dynamic weighting and computational issues in heuristic problem solving. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, 12–17.

Sturtevant, N. 2012. Benchmarks for grid-based pathfinding. *Transactions on Computational Intelligence and AI in Games* 4(2):144 – 148.

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

Thayer, J. T., and Ruml, W. 2008. Faster than Weighted-A*: An optimistic approach to bounded suboptimal search. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 355–362. AAAI.

Vemula, A.; Mülling, K.; and Oh, J. 2016. Path planning in dynamic environments with adaptive dimensionality. In *Symposium on Combinatorial Search (SoCS)*, 107–116.

Yahja, A.; Stentz, A.; Singh, S.; and Brumitt, B. 1998. Framed-quadtree path planning for mobile robots operating in sparse environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, 650–655.