Multi-Resolution A*

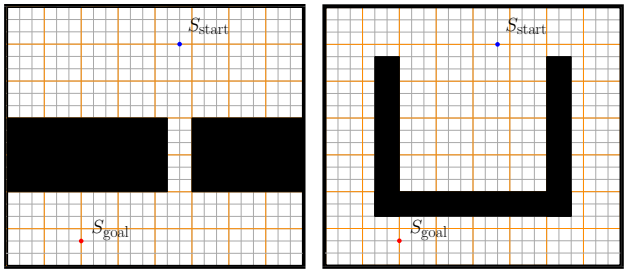
Abstract

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 exponentially 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 with 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 of other 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 by which each dimension is discretized and the size of the environment or the map (Elbanhawi and Simic 2014). The size of state spaces increase exponentially with the increased dimension as well as the resolution.

Consider a very large map most of which is free space, yet it has a number of narrow passages, which the planner has to plan through. Fig. 1 shows two snippets from this map discretized by two resolution levels. For the example shown in Fig. 1(a), a search on such a domain with the coarse resolution space will fail since the narrow passage enforces the requirement of having a high resolution for completeness sake. Generally, not only does a low resolution space weakens the completeness guarantee, but also sacrifices solution quality.



(a) narrow_passage (b) local_minima

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

Consider another region of this example map shown in Fig. 1(b). To find a path from the shown start to goal state, 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. Generally, 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* employs 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. We borrow ideas from Multi-Heuristic A* (MHA*) algorithm (Aine et al. 2016); we provide completeness and bounded suboptimality guarantees by using an anchor search which runs on a particular resolution space. MHA* uses multiple potentially inadmissible heuristics in addition to a single consistent anchor heuristic. Instead of taking advantage of multiple heuristics in different searches, we utilise multiple resolution state spaces for different searches. We prove that our algorithm is resolution complete with respect to 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, where MRA* is compared with other search-based algorithms and sampling-based algorithms. The results verify that MRA* outperforms other algorithms in several aspects.

2 Related Work

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

Sampling-based methods are popular candidate for high-dimensional motion planning problems. 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 and state lattices. Moore et al. came up with the Parti-game algorithm (Moore and Atkeson 1995), which adaptively discretizes the map with high resolutions at the border between obstacles and free space and low resolutions 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 graph edges. These methods are not applicable to high-dimensional spaces due to the required memory footprint.

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. This relies on the condition that there is a smooth transition between high-resolution and low-resolution lattices. This approach constrains the search to different resolutions spatially which weakens the completeness guarantee.

Another class of methods for reducing the size of search space is planning in non-uniform state dimension and action 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. Their method provides guarantees on completeness but not optimality.

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. In addition to these searches MRA* uses an anchor search which is an optimal A* search, to provide bounds on the solution quality. We propose two variants of the algorithm which differ in the representation of the anchor search and discuss their properties.

3.1 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} 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 it 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. $\textsc{Succs}(s, i)$ returns all successors of s with respect to resolution i. $\textsc{GetSpaceIndices}(s)$ returns the 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

Algorithm 1 Multi-Resolution A*

```
1:  procedure MAIN
2:      g(s_start) = 0; g(s_goal) = ∞
3:      bp(s_start) = bp(s_goal) = null
4:      for i = 0, ..., n do
5:          OPEN_i ⟵ ∅
6:          CLOSED_i ⟵ ∅
7:          Insert s_start in OPEN_i with KEY(s, i)
8:      while OPEN_0 ≠ ∅ do
9:          i ⟵ CHOOSEQUEUE( )
10:         if OPEN_i.MINKEY() ≤ ω * OPEN_0.MINKEY() then
11:             if g(s_goal) ≤ OPEN_i.MINKEY() then
12:                 Return path pointed by bp(g(s_goal))
13:             else
14:                 s = OPEN_i.Pop()
15:                 EXPANDSTATE(s, i)
16:                 Insert s into CLOSED_i
17:         else
18:             if g(s_goal) ≤ OPEN_0.MINKEY() then
19:                 Return path pointed by bp(g(s_goal))
20:             else
21:                 s = OPEN_0.Pop()
22:                 EXPANDSTATE(s, 0)
23:                 Insert s into CLOSED_0
```

with the priority function $\text{KEY}_i(s)$ and a list of expanded states CLOSED_i. Additionally, each queue has a function $\text{OPEN}_i.\text{MINKEY}()(s)$ which returns the minimum KEY value for the ith queue.

3.2 Algorithm

The main algorithm is presented in Alg. 1. The lines 2- 7 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 the corresponding priority values.

The algorithm runs until OPEN_0 gets empty (line 8) or any of the two termination criteria (lines 11 or 18) are met. In line 9, 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 on Line 10 controls the inadmissible expansions from other queues by expanding from the anchor whenever the condition fails. The expansions from the anchor queue monotonically increase $\text{OPEN}_0.\text{MINKEY}()(s)$ 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 $\text{EXPANDSTATE}(s, i)$ function "partially" expands state s in the search i by executing actions only in 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.

[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.

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) + εh(s)
6:  procedure EXPANDSTATE(s, i)
7:      for all s' ∈ SUCCS(s, i) do
8:          if s' was never generated then
9:              g(s') = ∞; 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' ∉ CLOSED_0 then
13:                 if 0 ∈ GETSPACEINDICES(s') then
14:                     Insert/Update s' in OPEN_0 with KEY(s', 0)
15:                 for each i ∈ GETSPACEINDICES(s') - {0} do
16:                     if s' ∉ CLOSED_i then
17:                         Insert/Update s' in OPEN_i with KEY(s', i)
```

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 resolution (see lines 13 and 15).

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.*

Proof. If s is expanded by the search i it is inserted in CLOSED_i (Alg. 1, Line 16) 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.*

This holds true because the algorithm terminates until it finds a solution or the anchor state space is exhausted (Alg. 1, line 8)

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 19 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 12), then from lines 10 and 11 we have,

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

□

3.4 A variant of MRA*

Instead of having a single resolution space for the anchor search, we could have 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 3D Dijkstra's from the goal end-effector's position. We used Euclidean distance as cost function for 2D and 3D, and Manhattan distance for 7D. For all the domains, the anchor search of MRA* is set to be in the highest resolution search 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 have a plot showing improvements of MRA* over baselines, where improvements are computed as the average metric values of baselines divided by that of MRA*'s. For these plots we only report results for common success tests. Besides, we also show tabulated results for all the metrics.

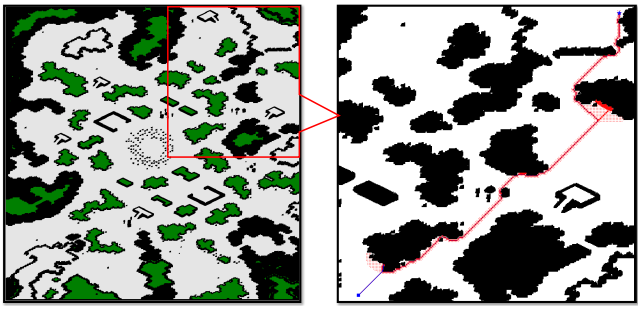


Figure 2: A 2D solution example. The planner is planning from start (square) to goal (star). The red dots are expanded states and the blue line is the solution returned by planner.

4.1 2D Space Planning Results

Domain: We have 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 7x and 21x 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 then compare our algorithm WA* with Multiple Resolutions (WA-MR), WA* with highest resolution (WA-High) and with lowest resolution (WA-Low) respectively. WA-MR's action space uses the union of all the resolutions in a single queue. Besides, we compare our algorithm with static environment 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 value both to 3.0. For other search based algorithms, we set the weight to 3.0 as well, which enforces the same suboptimality bounds for all the algorithms.

Results and Analysis: The results of 2D planning are presented in Fig. 4(a) and Table. 1a. A test map and a sample solution from MRA* is shown in Fig. 2. It can be observed at the top-right region of the right figure, MRA* sparsely searched the local minimum region and exited swiftly. This is consistent with the behaviour what we described in Fig. 1(b).

As is shown in Fig. 4(a), our algorithm outperforms WA-MR and WA-High in speed and number of expansions. 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

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

escape local minimum while still using 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. Compared to QDTree, MRA* seems slower, since QDTree performs the search on a pre-constructed explicit graph which had an average pre-computation time of 36 seconds for the two maps. In quad-tree map discretization, large open spaces will not be further discretized into smaller units, this helps to keep the size of state space small. As a result, there is significantly smaller amount of expansions in QDTree search. Considering the averaged solution cost in Table. 1a, the quality of solutions from each algorithm is consistent except QDTree. This is because QDTree has very coarse discretization in free spaces, which results in weaker optimality bounds and hence slightly worse solution cost.

4.2 3D Space Planning Results

Domain: There are two different environments in 3D planning experiments, one of them is shown in Fig. 3. The other map contains outdoor scenes such as mountains and buildings etc. The maps are discretized to a grid of size $1000 \times 1000 \times 400$ cells in the highest resolution. 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. We ran experiments on MRA*, WA-MR, WA-High and WA-Low baselines. 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 in scene Fig. 3 are presented in Fig. 4(b). With the same branching factor, WA* in coarse resolution space is significantly faster and has far less expansions. As has been mentioned, this implementation is not complete and the optimality bounds in this discretization is slightly weaker, which leads to slightly worse solution qualities. Regarding planning time, MRA* is the fastest algorithm as sharing states between queues speeds up the search on higher resolution spaces. And in WA-MR, the branching factor reaches 78 because states are fully expanded, which hugely slowed down the search and furthermore lowered the success rate (see Table. 1b). MRA* also show considerable number of expansions because it doesn't waste resources expanding high resolution states in large open spaces. In terms of solution cost, MRA* generates solutions slightly worse than WA-MR and WA-High, but still bounded.

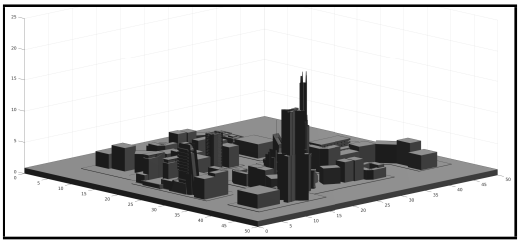


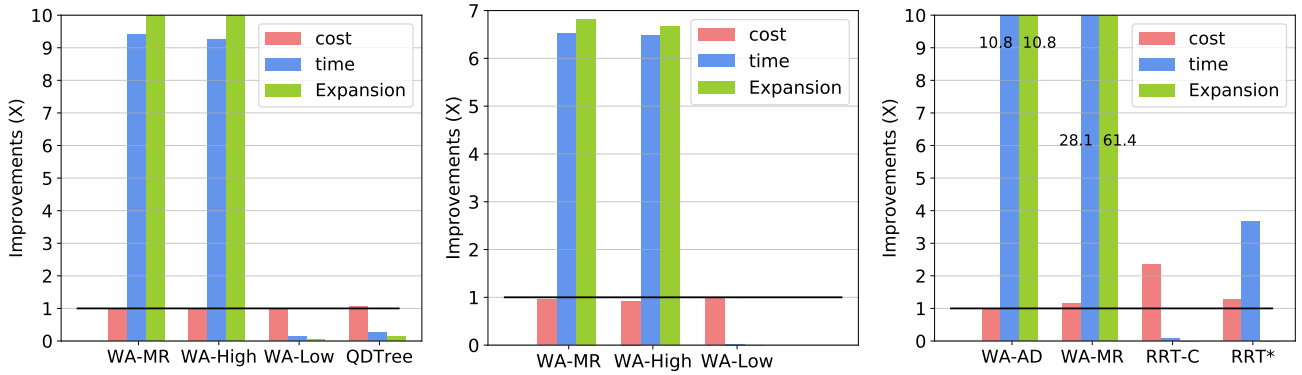
Figure 3: A mesh model of city used as a planning scene for 3D planning.

4.3 7D Space Planning Results

The 7D space planning is implemented in simulation provided by SMPL[2] library and Robot Operating System (ROS).

Domain: In this category, we have single-arm motion planning problems. There are 4 different planning scenes presented in Fig. 5. The start and goal pairs are randomly generated. There are 70 trails for each scene. The cost function is then defined as the sum of angular distance traveled by all joints, in rad. In all baseline programs, the same collision checker is provided. We have RRT-Connect (RRT-C), RRT* as our sampling-based algorithm baselines. In addition, we use WA-MR and WA* with adaptive dimensionality search (Kalin Gochev and Likhachev 2013) (WA-AD) as search-based planning baselines. The implementations

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



(a) The results of 2D planning.

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

(c) 7D planning in scene Industrial.

Figure 4: Improvements of MRA* over baseline algorithms

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} each action corresponds to a joint angle change of 27°. In M_{middle} the resolution of a joint angle change is 9°. In M_{high} the resolution is 3° per angle change. Except for static motion primitives, adaptive actions are generated online via inverse kinematics computation (Cohen et al. 2011) to *snap* end-effector to the goal pose when it is in the vicinity of goal pose.

Results and Analysis: Firstly, consider results of *Industrial* scene are presented in Fig. 4(c), plots of results in other scenes are consistent. In terms of planning time, MRA* outperforms other search-based baselines and RRT*. Considering planning time and number of expansions, surprisingly, there are even more than 10 times improvement in MRA* over WA-AD and WA-MR. That is, MRA* shows more advantages in higher dimensional spaces. With respect to solution cost, MRA* performs no worse than any other algorithms on common succeeded trails, which verifies that MRA* provides bounded suboptimal solutions.

Secondly, consider results documented in Table. 2. MRA* has the highest success rates in all scenes, which is even better than the RRT-Connect algorithm. Although MRA* is not the fastest algorithm, it is far faster than WA-AD and WA-MR. Regarding solution cost, as search-based algorithms are bounded-suboptimal algorithms, they consistently provides better solutions than the other two sampling-based algorithms, which also, demands less effort on post-processing the solutions. On the contrary, due to its random sampling feature, RRT-Connect returns low- and unbounded-quality solutions, which is far worse than MRA*. As the space size increases drastically with high dimensionality, WA-AD and WA-MR failed to find a solution within time limit. As WA-AD plans with adaptive dimensionality, the search is accelerated when it searches in lower dimensionality space, resulting in faster mean time and higher success rate than WA-MR. While WA-MR performs worse 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 is actually search in the hybrid graph, which potentially has better optimal bounds as mentioned in Section 3.4. More importantly, MRA* has the highest success rate over all planning domains and planning scenes.

5 Conclusion and Future Work

We presented a heuristic search-based algorithm that utilises multiple search spaces owing to 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 on the anchor resolution space. We show that algorithm show significant improvements over the baselines on large 2D, 3D domains and high-dimensional motion planning problems, more importantly in terms of success rates.

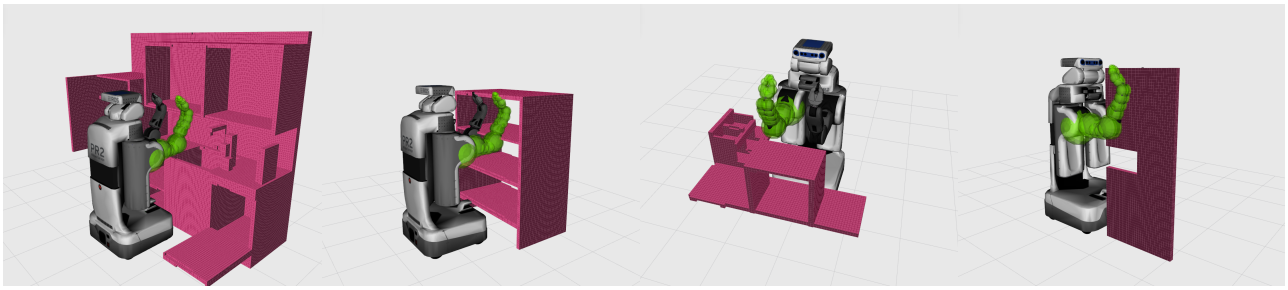
While the results are promising, we believe that there is scope for further improvements. Future research directions could be 1) using multiple heuristics within the different resolutions searches to speed up the search, 2) adding dynamic snap motions/primitives for efficient sharing between the different spaces, and 3) using a large ensemble of resolution spaces and optimizing for the scheduling policy.

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.

Brock, O., and Kavraki, L. E. 2001. Decomposition-based motion planning: A framework for real-time motion plan-



(a) Kitchen (b) Bookshelf (c) Industrial (d) Narrow Passage

Figure 5: The planning scenes of single-arm manipulation problem.

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

ning in high-dimensional spaces. In *IEEE International Conference on Robotics and Automation (ICRA)*, 1469–1474.

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 partigame 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.

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.