Improved Roadmap Connection via Local Learning for Sampling Based Planners

Chinwe Ekenna, Diane Uwacu, Shawna Thomas, Nancy M. Amato

Abstract— **Probabilistic Roadmap Methods (PRMs) solve the motion planing problem by constructing a roadmap (or graph) that models the motion space when feasible local motions exist. PRMs and variants contain several phases during roadmap generation i.e., sampling, connection, and query. Some work has been done to apply machine learning to the connection phase to decide which variant to employ, but it uses a global learning approach that is inefficient in heterogeneous situations.**

We present an algorithm that instead uses *local* learning: it only considers the performance history in the vicinity of the current connection attempt and uses this information to select good candidates for connection. It thus removes any need to explicitly partition the environment which is burdensome and typically difficult to do. Our results show that our method learns and adapts in heterogeneous environments, including a KUKA youBot with a fixed and mobile base. It finds solution paths faster for single and multi-query scenarios and builds roadmaps with better coverage and connectivity given a fixed amount of time in a wide variety of input problems. In all cases, our method outperforms the previous adaptive connection method and is comparable or better than the best individual method.

I. Introduction

Motion planning is a well studied problem in robotics that consists of finding a path from the start to the goal while avoiding collisions with objects in the environment. Motion planning solutions have been applied in several areas of medicine, gaming, and virtual reality simulation [18], [24]. Sampling based motion planning algorithms, such as Probabilistic Roadmap Methods (PRMs) [13] and Rapidly-Random Exploring Trees (RRTs) [16], are widely used to solve a variety of problems.

PRMs generate nodes representing the robot's configuration space, connect them, and produce a graph containing feasible trajectories. RRTs use local exploration by progressively expanding a tree outwards in random directions until an input query can be solved, or a maximum number of iterations has been reached.

This research supported in part by NSF awards CNS-0551685, CCF-0833199, CCF-0830753, IIS-0917266, IIS-0916053, EFRI-1240483, RI-1217991, by NSF/DNDO award 2008-DN-077-ARI018-02, by NIH NCI R25 CA090301-11, by DOE awards DE-FC52-08NA28616, DE-AC02-06CH11357, B575363, B575366, by THECB NHARP award 000512-0097-2009, by Samsung, Chevron, IBM, Intel, Oracle/Sun, by Award KUS-C1-016-04, made by King Abdullah University of Science and Technology (KAUST), by NSF broadening participation in computing program (NSF CNS-0540631) and by the Schlumberger Faculty for the Future Fellowship. This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

C. Ekenna, D. Uwacu, S. Thomas, and N. M. Amato are with the Department of Computer Science and Engineering, Texas A&M University, College Station, TX, 77843, USA. {cekenna, sthomas, amato}@cse.tamu.edu {diane.uwacu}@eagles.oc.edu

Many methods exist for the various tasks involved, but selecting the best one for a particular input problem is difficult. This issue is only magnified in heterogeneous environments where different algorithmic choices may apply for different regions. Making good choices for each of the algorithmic steps in these techniques remains difficult. Thus, machine learning approaches have been used to dynamically decide which approaches to take for generating samples (e.g., Hybrid PRM [10]) and connecting them together (e.g., Adaptive Neighbor Connection (ANC) [8]). Both of these approaches learn globally and only perform well in homogeneous spaces or partitioned spaces with homogeneous partitions.

In this work, we select better connection candidates via *local* learning. We present a new method that removes the need to explicitly partition the environment by localizing learning to within the vicinity of the current configuration being connected to the roadmap. A connection method adds edges to the roadmap by selecting pairs of nodes to connect, and using a local planner, computes a trajectory, adding the edge to the roadmap if valid. Distance metrics and local planners are two key inputs to a connection method. When choosing a connection method to connect a given configuration to the roadmap, we dynamically determine a neighborhood around that configuration.

This neighborhood is defined as the set of nearest neighbors as identified by some distance metric. Using a fixed local planner and sampling method, we utilize the performance history of only those connection attempts within this neighborhood to bias learning. Thus, our method adapts spatially and temporarily, and no prior knowledge about connection methods involved is needed.

We show the impact of localized learning, and apply it to various heterogeneous environments to demonstrate the generality of our approach. Our contributions include:

- A locality based learning method that selects better connection candidates,
- improved performance in connecting and building roadmaps for heterogeneous problems without explicit partitioning of the environment and,
- results demonstrating a reduction in planning time, an increase in number of queries solved, and a better utilization of available connection methods.

We run experiments on many different problems including rigid bodies in 2D and 3D environments, multiple robots, an articulated robot with 36 degrees of freedom (DOF), and a real world simulation of the KUKA youBot robot [14]. We study both single query and multi-query scenarios in heterogeneous environments. In single query scenarios,

our method improves on the time required to construct a query-solving roadmap. In multi-query scenarios, our method solves a high percentage of possible queries. We show that localized learning always outperforms global learning, including RRT methods, and is vital in heterogeneous environments, including real world scenarios.

II. Related Work

In this section we discuss PRMs, nearest neighbor selection methods, and adaptive motion planning frameworks including Adaptive Neighbor Connection (ANC) [8].

A. Probabilistic RoadMaps for Motion Planning

PRMs [13], a sampling-based motion planning approach, use a two stage process: roadmap construction and query processing. During roadmap construction, PRMs sample the configuration space (i.e., set of all possible robot placements (C-Space)), retain valid ones as roadmap nodes, and attempt to connect them using some local planner. The probability of failing to find a path when one exists decreases exponentially with the number of configurations in the roadmap [12].

There have been a number of methods proposed for locating candidate neighbors for connection because it is intractable to simply attempt all possible connections (i.e., $O(n^2)$ attempts). In [9], the maximal connectivity achieved by different sampling methods was compared to that of the C-Space. Maximal connectivity ensures that if a path between the start and the goal configuration exists, then this path can be found in the roadmap. They describe the properties of many neighbor finding approaches and motivate research on connections based upon reachability analysis. However, this work relies on a C-Space discretization and cannot be applied to high DOF problems.

B. Nearest Neighbor Selection

Nearest neighbor searching is a fundamental and important component in sampling-based motion planning [7]. Planners use nearest neighbor search data structures to find and connect configurations in order to compute a motion plan.

Nearest neighbor searching is often a performance bottleneck, particularly when the dimensionality of the space increases [11]. It is sometimes desirable to sacrifice accuracy for speed by using approximate methods. Some approximate methods use data structures to more efficiently compute nearest neighbors. *Metric Trees* [25] organize the configurations in a spatial hierarchical manner by iteratively dividing the set into two equal subsets resulting in a tree with $O(\log n)$ depth. However, as the dataset dimensionality increases, their performance decreases [17]. *KD-trees* [3] extend the intuitive binary tree into a D-dimensional data structure which provides a good model for problems with high dimensionality. However, a separate data structure needs to be stored and updated each time a node is added to the roadmap. Other approximate neighbor finding methods include spill trees [17], MPNN [26], and Distance-based Projection onto Euclidean Space [22]. These methods can dramatically reduce computation time for nearest neighbor searches, but there is no proof of optimality for asymptotically optimal motion planners when using these approximate searches.

We focus on exact k nearest neighbors given by some distance metric. We also use two useful variants [19]: K-Closest,K-Rand and R-Closest,K-Rand. K-Closest,K-Rand randomly selects k neighbors from the k_2 closest nodes, where typically $k_2 = 3k$. R-Closest,K-Rand selects k random neighbors from those within a distance r specified by some distance metric. While we only study exact methods here, approximate methods may be used.

C. Distance Metrics

A distance metric is a function δ that computes some "distance" between two configurations $a = \langle a_1, a_2, \ldots, a_d \rangle$ and $b = \langle b_1, b_2, \ldots, b_d \rangle$, i.e., $\delta(a, b) \rightarrow \mathbb{R}$, where d is the dimension of a configuration. A good distance metric for PRMs predicts how likely a pair of nodes can be connected. In this paper, we study two different distance metrics:

Euclidean: The Euclidean distance metric gives equal weighting for all dimensions:

$$\delta(a, b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_d - b_d)^2}$$

The scaled Euclidean distance metric is a variant

$$\delta(a, b) = \sqrt{s(\text{pos\_mag})^2 + (1 - s)(\text{ori\_mag})^2}$$

where pos_mag is the Euclidean distance of the positional dimensions, ori_mag is the Euclidean distance of orientational dimensions, and s is a weighting parameter. It is cheap and adequate in many situations. However, it is not a good predictor when the local planner differs from a simple straight line. Here, we use the scaled euclidean distance metric with $s = 0.5$ and refer to it as "ScaledEuclidean". Note that this is equivalent to Euclidean distance for robots without articulation.

Swept Volume: Swept volume is the volume generated by the continuous motion (translation and/or rotation) of a geometric object through space. The swept volume distance is the volume swept by the robot while following the motion prescribed by the local planner. For an articulated linkage, this becomes the sum of the swept volumes of each of the links. This distance measure is expensive but accurate for any local planner. It is referred to as "LpSwept" in this work.

D. Adaptive Motion Planning Frameworks

Many techniques use machine learning to improve the performance of sampling-based motion planning. In this section we briefly highlight some of these methods.

1) Feature Sensitive Motion Planning [20]: uses machine learning to partition and characterize planning problems. Here, the planning space is recursively subdivided. Each region is classified and assigned an appropriate sampling method. This approach is able to map workspace/C-Space topologies for a particular sampling method to generate configurations in. However, it is not able to adapt sampling methods over time. It also does not consider different connection methods in the regions or which pairs in a region (or between regions) are good connection candidates.

2) HybridPRM [10]: employs a reinforcement-learning approach to select a sampling method that is expected to be effective at the current time in the planning process. These sampling methods are applied globally over the whole problem, and features of the planning space, such as topology, are not used when deciding where to apply them. They also do not adapt different connection methods.

3) Utility Guided Sampling [5], [6]: uses information from previous experiences to guide sampling to more relevant areas of C-Space. Every exploration of C-Space provides information to the motion planner. They construct an approximate model of C-Space. Their model captures and maintains information from each configuration and predicts the state of unobserved configurations to reduce collision detection calls. This approach is only focused on sampling.

4) Workspace-based Connectivity Oracle (WCO) [15]: uses domain knowledge (i.e., workspace geometry and sampling history) to adaptively sample the workspace. WCO is composed of many component sampling methods, each based on a geometric feature of a robot. It uses adaptive hybrid sampling to combine information from workspace geometry and sampling history. This work improves narrow passage connectivity by generating configurations with a high probability of getting chosen.

5) RESAMPL [23]: uses local region information to make decisions about how and where to sample and which samples to connect together. This use of spatial information about the planning space enables RESAMPL to increase sampling in narrow regions and decrease sampling in free regions.

6) Instance-Based Learning [21]: uses historical information from collision calls to compute an approximate C-Space model via a hash table. An instance-based learning algorithm replaces expensive collision-checking with a probability-based prediction of the sample's validity. Even though this approach is efficient, it might be misleading in narrow passages where a point in the free space might have all its neighbors in obstacle space and be mistakenly rejected.

7) Learning from Experience [4]: proposes a framework called Lightning that is able to learn from experience. Lightning consists of two modules that run in parallel: a planning from scratch module and a module that retrieves and repairs paths stored in the path library. Any path that is generated for a new query is checked by a library manager to decide how expensive the path is and how similar it is to previously generated paths. However, as the size of the library gets bigger, it becomes impractical to add new paths.

E. Adaptive Neighbor Connection (ANC)

The work in [8] adaptively selects the appropriate connection method to use over time. Several popular connection methods are put together, and their performance is recorded and used to decide which one should be employed to connect new configurations to the existing roadmap. It does this by maintaining a selection probability for each method.

Specifically, ANC first finds a probability p_i^* for each connnection method cm_i ignoring the cost:

$$p_i^* = (1-\gamma)\frac{w_i(t)}{\sum_{j=1}^{m} w_j(t)} + \gamma\frac{1}{m}, i = 1, 2, ..., m, \qquad (1)$$

where $w_i(t)$ is the weight of cm_i in step t, t is the number of connection attempts made by the planner, γ a fixed constant that represents the randomness of the connection method choice, and m is the number of connection methods in the set. This formula computes the probability p_i^* as a weighted sum of the relative weight of the cm_i and the uniform distribution, which guards against starvation.

Connection methods are rewarded according to the number of returned configurations that result in successful connections. The reward is updated on the cost insensitive probability because it should be independent of the accrued cost. After finding the updated reward, the weight is calculated as a function of the updated reward:

$$w_i(t+1) = w_i(t)\exp\frac{\gamma x_i^*}{m}, i = 1, 2, ..., m, \qquad (2)$$

where x_i^* is the updated reward found by dividing the reward by the cost insensitive probability. For the weights to adapt quickly, ANC uses an exponential factor.

Finally, ANC calculates a cost sensitive probability by including the cost of connection attempts:

$$p_i = \frac{\frac{p*_i}{c_i}}{\sum_{j=1}^{m}\frac{p_j^*}{c_j}}, i = 1, 2, ..., m. \qquad (3)$$

Higher cost methods have fewer chances of being selected.

The main weakness of this approach is that it bases decisions on the connection method's performance over the entire environment. This becomes especially problematic in heterogeneous environments. To obtain better results, it became necessary to first partition the environment into smaller regions that are likely to be homogeneous [8]. However, this puts greater burden on the user, particularly as the complexity of the problem increases, and a poorly selected partitioning could be quite detrimental to learning.

III. Spatial learning Approach

In this work, we present a new method called Spatial Adaptive Neighbor Connection (ANC-Spatial) that localizes learning to within the vicinity of the current configuration being connected. When choosing a connection method to connect a given configuration to the roadmap, we dynamically determine a neighborhood around that configuration. We define this neighborhood as the set of nearest neighbors identified by some distance metric. We use the performance history of only those connection attempts within this neighborhood for learning. Thus learning is continuous and localized. Our method adapts both spatially and temporarily with no prior knowledge about the connection methods needed.

Figure 1 shows an example 2D point robot environment that is largely free with a small cluttered region in the bottom

left corner. The roadmap is constructed with two candidate connection methods: CM_A (blue) and CM_B (red). Overall, the most successful connection method is CM_A (indicated by a greater number of blue edges). However, in the cluttered region in the bottom left, CM_B is much more successful. When connecting node q (in green) to the roadmap, it is important to take locality into account. A global learning method, such as ANC, would select CM_A, but this would be a poor choice. A local learning method, such as ANC-Spatial, would instead wisely choose CM_B because CM_B is much more successful in this local area.

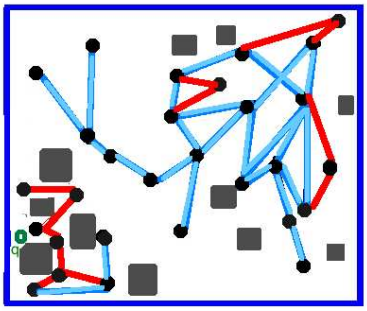


Fig. 1: A 2D heterogeneous environment with a point robot. Two connection methods are used: CM_A (blue) and CM_B (red). When connecting q (in green), it is important to learn from local information as performance is non-uniform.

Algorithm 1 describes ANC-Spatial in the context of PRMs. It takes as input a vertex q, any number of connection methods, a locality neighbor finder Nf_{local}, the local planning method lp, and the graph G. We start by initializing the selection probability distribution P_q with the uniform distribution. We then find the neighbors to q using Nf_{local} and store them in N_{init}. The neighborhood identified by N_{init} defines the dynamically determined learning region. For every configuration $n \in N_{init}$, we update P_q using Equation 3 with the connection method, reward, and cost history associated with it. We then randomly select a connection method to connect q to G based on P_q. We add the edge (q, n) if the local planner lp returns true. Finally we append q and n with the connection method tag, reward, and cost.

IV. Experimental Results

We compare the performance of ANC-Spatial with ANC [8] and individual connection methods in two types of popular motion planning scenarios: single query runs and multi-query situations. For the single query case, we examine the time to solve a query. For the multi-query case, we look at the roadmap coverage and connectivity (as measured by the number of queries solved). It is important to note that ANC and ANC-Spatial are the only two methods that learn during the connection phase of PRM (other related work focuses on sampling only) and are thus the methods studied here.

Algorithm 1 ANC-Spatial

Input. A connecting vertex q, a set of connection methods CM, a locality neighbor finder Nf_{local}, a local planner lp and a graph G.

Output. A graph G with additional edges to/from q.

1: Initialize a set of connection method probabilities P_q to the uniform distribution
2: Initialize N_{init} = the set of neighbors to q using Nf_{local}
3: **for** each $n \in N_{init}$ **do**
4: Update P_q using Equation 3 and all $< cm, reward, cost >$ tuples stored in n
5: **end for**
6: Randomly select a cm_q according to P_q
7: $N = cm_q$.FIND_ NEIGHBORS(q, G)
8: **for** each $n \in N$ **do**
9: **if** lp.IS_CONNECTABLE(q, n) **then**
10: G.ADD_EDGE(q, n)
11: **end if**
12: Append q and n with $< cm_q, x_q, c_q >$ where x_q is the reward and c_q is the cost of the connection attempt
13: **end for**

We look at a variety of input problems including 2D and 3D environments (see Figure 2), single robots and multiple robots, rigid body robots and robots with up to 36 DOF. We also study ANC-Spatial performance in two real world simulations. The environments are heterogeneous, and we show that ANC-Spatial solves the motion planning problem without having to explicitly partition the environment while other methods suffer from applying a global scheme.

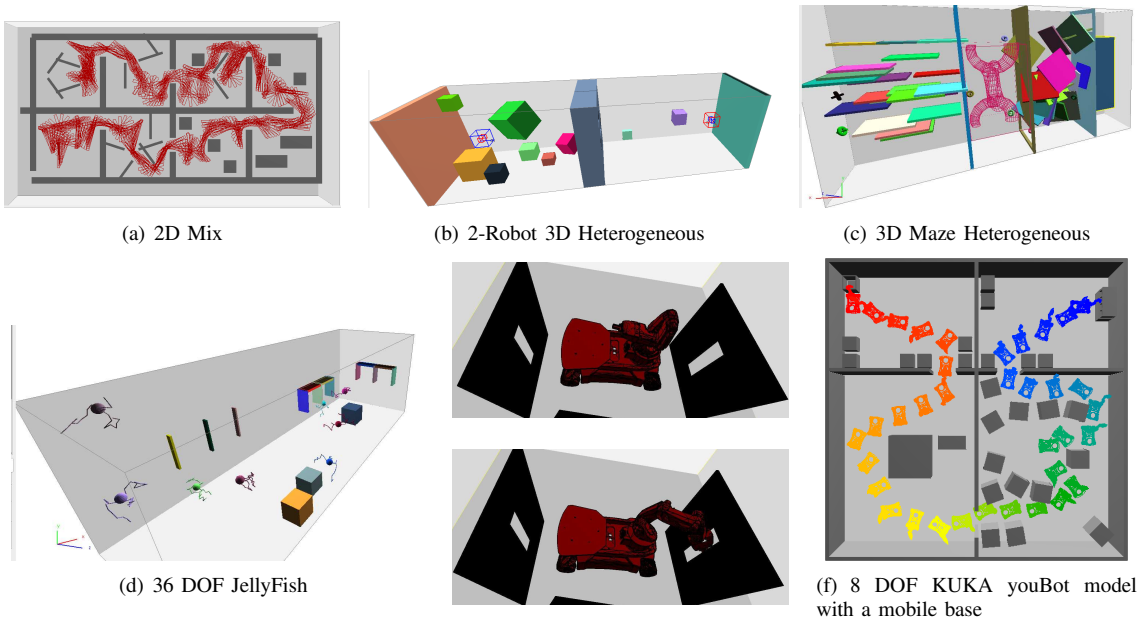
A. Experimental Setup

We compare ANC-Spatial, ANC, and four individual connection methods: K-Closest using ScaledEuclidean and LpSwept, K-Closest,K-Rand using ScaledEuclidean, and R-Closest,K-Rand using ScaledEuclidean with $k = 10$, $k_2 = 3k$, and r as the average pairwise distance among a sample set of configurations. For ANC-Spatial, Nf_{local} is the 10 nearest neighbors by ScaledEuclidean which resulted in the best performance on preliminary experiments.

Our tests were run in a C++ motion planning library developed in the PARASOL lab at Texas A&M. All experimental results are averaged over 10 runs with different random seeds. For all our experiments we used Obstacle Based PRM [1] for sampling. We use the StraightLine local planner and the RotateATS local planner [2] with $s = 0.5$. RotateATS attempts to find a path by translating from configuration q_A to q_B the portion of the distance denoted by s, rotating about its axis, and then translating the remaining way to q_B.

B. Single Query Results

Here we show the performance in single query scenarios. Roadmaps are incrementally constructed until the query is solved. We also compare against RRT which have been successful in single-query problems. We measure performance



(a) 2D Mix

(b) 2-Robot 3D Heterogeneous

(c) 3D Maze Heterogeneous

(d) 36 DOF JellyFish

(e) 8 DOF KUKA youBot model with a fixed base showing start and goal

(f) 8 DOF KUKA youBot model with a mobile base

Fig. 2: Environments studied.

as the time to solve the query and compare results using the various connection methods, ANC, ANC-Spatial and RRT.

1) 2D Mix Environment: The 2D Mix environment (see Figure 2(a)) has a long rectangular rigid robot in a heterogeneous environment containing 8 different rooms of different types including cluttered, free, and blocked regions. The start is at the bottom left, and the goal is located at the top left. The robot must traverse each room to solve the query.

Figure 3 shows the time to solve the query for each method studied in the context of StraightLine local planning (red bars and y axis on the left) and RotateATS local planning (green bars and y axis on the right). For StraightLine, R-Closest,K-Rand solves the query in the least amount of time followed by ANC-Spatial. LpSwept performs the worst because of its high computational cost. LpSwept is a more accurate method but due to the area covered during a sweep of the robot, such accuracy is not needed for StraightLine local planning. We also see that there is more time overhead needed to solve the query using RRT in comparison to both ANC and ANC-Spatial. Note that ANC-Spatial does better than ANC because the environment is heterogeneous, and there are no explicit subdivisions. Thus, spatial learning is needed.

With RotateAtS, we see a change in performance of the individual connection methods. Specifically, we see that LpSwept performs better here than before. ScaledEuclidean performs worse than LpSwept because it does not predict well the feasibility of RotateATS as it over-penalizes candidates with large rotation differences. ANC-Spatial was able to take advantage of this change in performance and clearly outperforms all the other methods. Again, ANC-Spatial outperforms ANC because the of issue of heterogeneity.

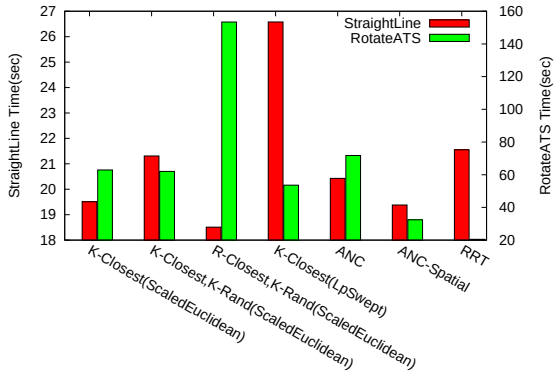


Fig. 3: Running time in the 2D Mix Heterogeneous environment using different local planners averaged over 10 runs.

2) 2-Robot 3D Heterogeneous Environment: We next study a higher dimensional problem (see Figure 2(b)). We have two rigid robots, a square robot and a smaller rectangular robot, and the aim is to interchange robot positions between the two heterogeneous rooms (one cluttered and one free) connected by a wall with a tunnel.

Figure 4 shows the time to solve the query for each method studied in the context of StraightLine local planning (red bars) and RotateATS local planning (green bars). Using StraightLine local planning, ANC-Spatial outperforms all other methods including the best connection method in this environment (i.e., K-Closest(ScaledEuclidean)). ANC-Spatial also outperforms ANC here as expected since we did not partition the environment. We also see that the time needed to solve the query using RRT is compara-

ble with ANC and worse than ANC-Spatial. The results for R-Closest,K-Rand(ScaledEuclidean) and K-Closest,K-Rand(ScaledEuclidean) indicate that introducing randomness is detrimental since the large dividing wall makes many node pairs poor choices for connection.

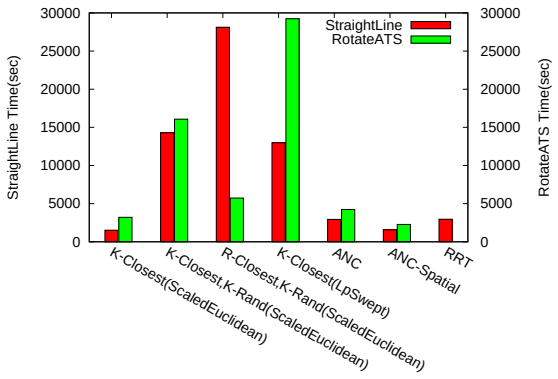


Fig. 4: Running time in the 2-Robot 3D Heterogeneous environment using different local planners averaged over 10 runs.

With RotateATS local planning, we see the performance of K-Closest(ScaledEuclidean) deteriorate, but ANC-Spatial still performs better than the other methods, including ANC.

The performance of ANC-Spatial in single query problems is indicative of the stability of our approach. Varying the connection method helps improve overall PRM performance. Spatial learning also plays a big part as ANC-Spatial outperformed ANC in every scenario.

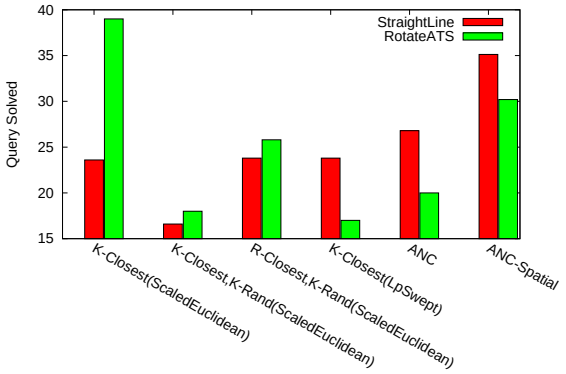
C. Multi-Query Experiments

In this section, we show the performance of the various methods in multi-query scenarios. Roadmaps are constructed within a fixed amount of time. We then count how many queries from a set of random samples are solvable by the resulting roadmap. We also look at the percentage of roadmap nodes contained in the largest connected component. This gives us indicators for coverage and connectivity.

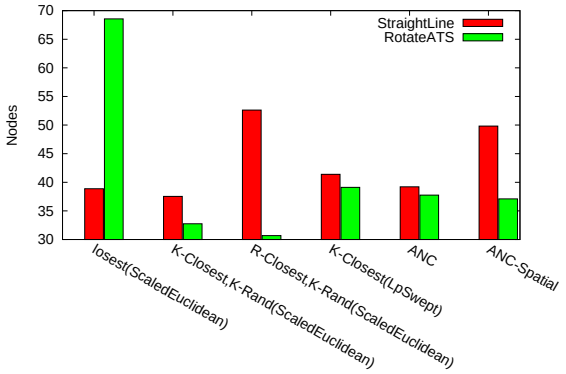
1) 3D Maze Heterogeneous Environment: A spherical robot must traverse 4 rooms separated by walls (see Figure 2(c). These rooms comprise of very narrow, cluttered, and maze regions. We allow roadmap construction to take 1200 seconds and provide 8 samples spread uniformly for querying. We then measure performance as the percentage of possible queries solvable from these samples by the roadmap.

Figure 5(a) (bars in red), shows the percentage of queries solved and Figure 5(b) (bars in red) the percentage of nodes in the largest connected component for each method using the StraightLine local planner. ANC-Spatial solves more queries than the other methods by almost a factor of 2. We see comparably higher percentage of nodes in its largest connected component. ANC-Spatial outperforms ANC due to the absence of explicit partitioning.

Figure 5 (bars in green) shows the results using the RotateATS local planner. ANC-Spatial is the second best in



(a) Percentage of Queries Solved



(b) Percentage of Nodes in the Largest Connected Component

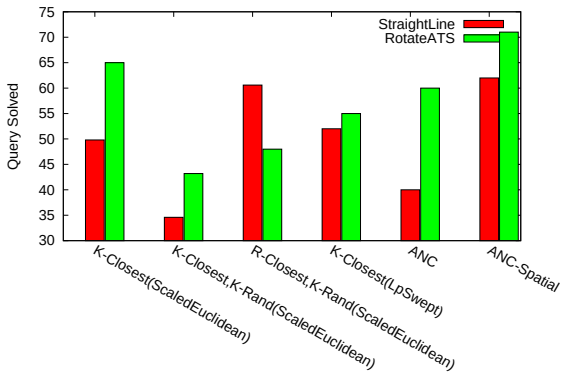
Fig. 5: Results for the 3D Maze Heterogeneous environment using different local planners averaged over 10 runs.

terms percentage of queries solved. Individual connection method performance has changed with the different local planners, but ANC-Spatial still outperforms ANC.

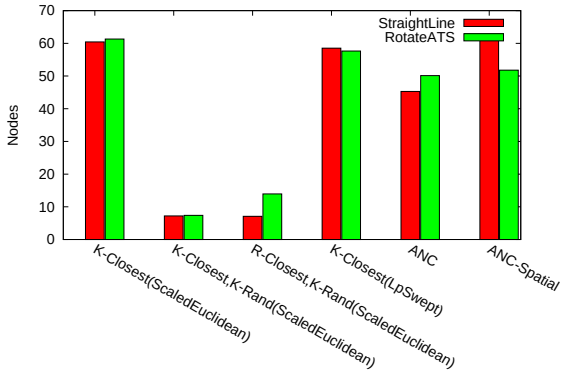
2) 36 DOF JellyFish Environment: Figure 2(d) shows a 36 DOF JellyFish environment where the robot has three articulated arms attached to a sphere where each arm has 10 links. The aim of this experiment is to see the performance of ANC-Spatial on a highly articulated and complex robot. Different types of obstacles are placed throughout the environment. We again provide 8 query samples and count how many possible queries can be solved. Roadmap construction is allowed to take 1500 seconds.

Figure 6 (bars in red) shows the percentage of queries (6(a)) and the percentage of nodes in the largest connected components (6(b)) using the StraightLine local planner for each method. ANC-Spatial again solves the greatest percentage of queries with a higher percentage of its nodes in the largest connected component. ANC-Spatial also outperforms ANC in this articulated linkage environment. This environment is prone to bad partitioning choices because a bad partitioning could easily form a narrow passage that cannot be traversed by this highly articulated linkage. Thus it is even more imperative that localized learning regions are dynamic.

Figure 6 (bars in green) shows results using the RotateATS local planner. ANC-Spatial also outperforms the other meth-



(a) Percentage of Queries Solved



(b) Percentage of Nodes in the Largest Connected Component

Fig. 6: Results for the 36 DOF JellyFish robot using different local planners averaged over 10 runs.

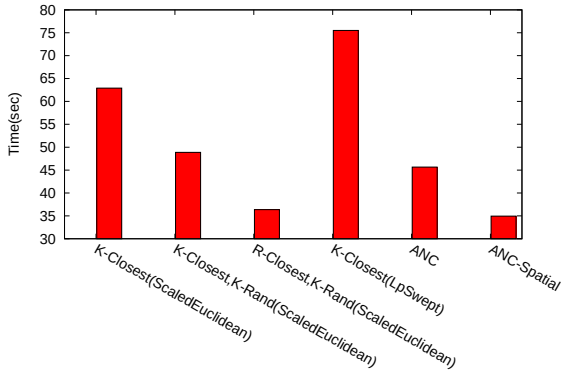
ods in terms of percentage of queries solved. We also have a good percentage of nodes in the largest connected component. Even a large change in local planners (as evidenced by changes in individual connection method performance) results in small change in ANC-Spatial performance.

D. Real World Simulation

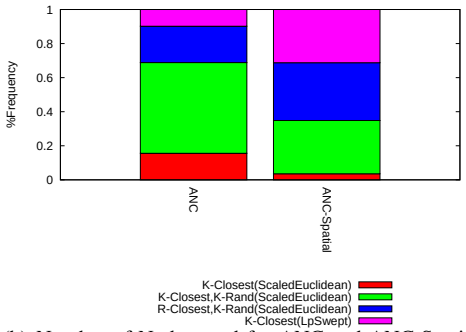
In this section, we show the performance of the various methods in real world simulation scenarios. We use an 8 DOF mobile manipulator in two difficult problems: navigating the arm in a highly constrained space with a fixed base and with a mobile base in an industrial setting that must traverse a cluttered space and reach into a cabinet. In both scenarios, we build roadmaps until the query is solved.

1) 8 DOF KUKA youBot model with a fixed base: Figure 2(e) is an 8 DOF KUKA youBot robot [14] with a fixed base. The robot has to reach a particular object placed in a narrow passage. By fixing the base, we reduce this robot to a 5 DOF problem. Figure 7(a) shows the time needed to get the end effector into the narrow window. ANC-Spatial requires the least time, including over ANC.

Figure 7(b) gives the usage frequency of the individual connection methods for ANC and ANC-Spatial. ANC-Spatial utilizes all the methods without being dominated by any connection method. ANC, however, suffers from poor choices due to the environment's heterogeneous nature. It learns and uses K-Closest,K-Rand(ScaledEuclidean) mostly and selects the worst performing method K-Closest(LPSwept) more than in ANC-Spatial.



(a) Time needed to Solve Query



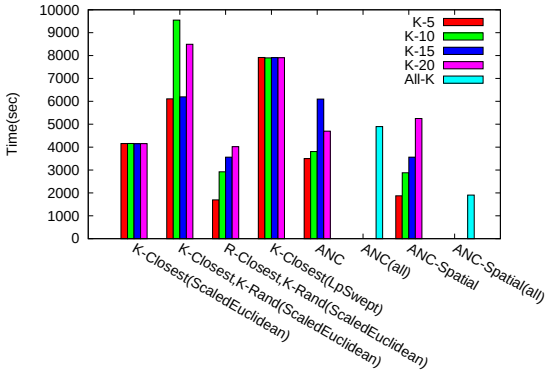
(b) Number of Nodes used for ANC and ANC-Spatial

Fig. 7: Results for the 8 DOF fixed base KUKA youBot Robot using StraightLine averaged over 10 runs.

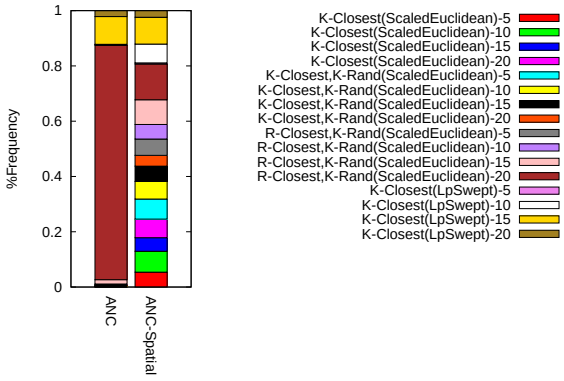
2) 8 DOF KUKA youBot model with a mobile base: Figure 2(f) shows an 8 DOF KUKA youBot robot with a mobile base that must pass through doorways while navigating around boxes in an industrial scene. The query requires the robot to move through each room and reach into a cabinet.

Figure 8(a) shows the time needed solve the query with different k values (5, 10, 15, 20) and the usage frequency of each individual connection method. K-Closest(Scaled Euclidean) and K-Closest(LpSwept) shows little change in the time needed with respect to k. R-Closest,K-Rand(Scaled Euclidean) is the best performing individual connection method. ANC-Spatial performs comparably for the different k values.

We also gave ANC and ANC-Spatial a chance to learn the appropriate k value by providing all instances of the connection methods as input. These are denoted as ANC(all) and ANC-Spatial(all) in Figure 8. ANC-Spatial(all) performs comparable to the best connection methods in the list and is still able to learn well when faced with different k values. ANC, however, has difficulty handling the large number of choices even though it utilizes the one of the best performing individual methods (see Figure 8(b)).



(a) Time needed to Solve Query



(b) Number of Nodes used for ANC and ANC-Spatial

Fig. 8: Results for the 8 DOF free base KUKA youBot Robot averaged over 10 runs.

V. Conclusion

We presented ANC-Spatial which uses localized learning to select appropriate connection methods during PRM roadmap construction. We performed experiments for 2D, 3D, high DOFs, and real world scenarios in heterogeneous environments. It performed better than ANC, a global learner, and was better than or comparable to the best performing connection method. ANC-Spatial also adapts to the different local planners. It removes the burden of deciding which method to use, leverages each method's strengths, and is extendable to include future methods.

References

[1] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo. OBPRM: an obstacle-based PRM for 3d workspaces. In *Proceedings of the third Workshop on the Algorithmic Foundations of Robotics*, pages 155–168, Natick, MA, USA, 1998. A. K. Peters, Ltd. (WAFR '98).

[2] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo. Choosing good distance metrics and local planners for probabilistic roadmap methods. *IEEE Trans. Robot. Automat.*, 16(4):442–447, August 2000.

[3] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998.

[4] D. Berenson, P. Abbeel, and K. Goldberg. A robot path planning framework that learns from experience. In *In the proceedings of the International Conference on Robotics and Automation (ICRA)*, 2012.

[5] B. Burns and O. Brock. Sampling-based motion planning using predictive models. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3313–3318, 2005.

[6] B. Burns and O. Brock. Toward optimal configuration space sampling. In *Proc. Robotics: Sci. Sys. (RSS)*, pages 105–112, 2005.

[7] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, June 2005.

[8] C. Ekenna, S. A. Jacobs, S. Thomas, and N. M. Amato. Adaptive neighbor connection for PRMs: A natural fit for heterogeneous environments and parallelism. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 1–8, November 2013.

[9] R. Geraerts and M. H. Overmars. Reachablility analysis of sampling based planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 406–412, 2005.

[10] D. Hsu, G. Sánchez-Ante, and Z. Sun. Hybrid PRM sampling with a cost-sensitive adaptive strategy. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3885–3891, 2005.

[11] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proc. ACM Symp. Theory of Computing (STOC)*, pages 604–613, 1998.

[12] L. E. Kavraki, J.-C. Latombe, R. Motwani, and P. Raghavan. Randomized query processing in robot path planning. In *Proc. ACM Symp. Theory of Computing (STOC)*, pages 353–362, May 1995.

[13] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.*, 12(4):566–580, August 1996.

[14] KUKA. http://www.youbot-store.com.

[15] H. Kurniawati and D. Hsu. Workspace-based connectivity oracle - an adaptive sampling strategy for prm planning. In *Algorithmic Foundation of Robotics VII*, pages 35–51. Springer, Berlin/Heidelberg, 2008. book contains the proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR), New York City, 2006.

[16] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *Int. J. Robot. Res.*, 20(5):378–400, May 2001.

[17] T. Liu, A. W. Moore, A. Gray, and K. Yang. An investigation of practical approximate nearest neighbor algorithms. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, pages 825–832, Cambridge, Massachusetts, 2005. MIT Press.

[18] D. Manocha, Y. Zhu, and W. Wright. Conformational analysis of molecular chains using nano-kinematics. *Computer Application of Biological Sciences (CABIOS)*, 11(1):71–86, 1995.

[19] T. McMahon, S. Jacobs, B. Boyd, L. Tapia, and N. Amato. Evaluation of the k-closest neighbor selection strategy for prm construction. Technical Report TR12-002, Texas A&M, College Station Tx., 2011.

[20] M. Morales, L. Tapia, R. Pearce, S. Rodriguez, and N. M. Amato. A machine learning approach for feature-sensitive motion planning. In *Algorithmic Foundations of Robotics VI*, Springer Tracts in Advanced Robotics, pages 361–376. Springer, Berlin/Heidelberg, 2005. (WAFR '04).

[21] J. Pan, S. Chitta, and D. Manocha. Faster sample-based motion planning using instance-based learning. In *Algorithmic Foundations of Robotics X*, volume 86 of *Springer Tracts in Advanced Robotics*, pages 381–396. Springer Berlin Heidelberg, 2013.

[22] E. Plaku and L. Kavraki. Quantitative analysis of nearest-neighbors search in high-dimensional sampling-based motion planning. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, July 2006.

[23] S. Rodriguez, S. Thomas, R. Pearce, and N. M. Amato. (RESAMPL): A region-sensitive adaptive motion planner. In *Algorithmic Foundation of Robotics VII*, pages 285–300. Springer, Berlin/Heidelberg, 2008. book contains the proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR), New York City, 2006.

[24] K. I. Tsianos, I. A. Sucan, and L. E. Kavraki. Sampling-based robot motion planning: Towards realistic applications. In *Computer Science Review*, volume 1, pages 2–11, 2007.

[25] J. K. Uhlmann. Satisfying general proximity/similarity queries with metric trees, 1991.

[26] A. Yershova and S. M. LaValle. Improving motion-planning algorithms by efficient nearest-neighbor searching. *IEEE Trans. Robot. Automat.*, 23(1):151–157, 2007.