2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Adaptive Neighbor Connection for PRMs: A Natural Fit for Heterogeneous Environments and Parallelism

Chinwe Ekenna, Sam Ade Jacobs, Shawna Thomas, Nancy M. Amato

Abstract— **Probabilistic Roadmap Methods (PRMs) are widely used motion planning methods that sample robot configurations (nodes) and connect them to form a graph (roadmap) containing feasible trajectories. Many PRM variants propose different strategies for each of the steps and choosing among them is problem dependent. Planning in heterogeneous environments and/or on parallel machines necessitates dividing the problem into regions where these choices have to be made for each one. Hand-selecting the best method for each region becomes infeasible. In particular, there are many ways to select connection candidates, and choosing the appropriate strategy is input dependent.**

In this paper, we present a general connection framework that adaptively selects a neighbor finding strategy from a candidate set of options. Our framework learns which strategy to use by examining their success rates and costs. It frees the user of the burden of selecting the best strategy and allows the selection to change over time.

We perform experiments on rigid bodies of varying geometry and articulated linkages up to 37 degrees of freedom. Our results show that strategy performance is indeed problem/region dependent, and our adaptive method harnesses their strengths. Over all problems studied, our method differs the least from manual selection of the best method, and if one were to manually select a single method across all problems, the performance can be quite poor. Our method is able to adapt to changing sampling density and learns different strategies for each region when the problem is partitioned for parallelism.

I. Introduction

The motion planning problem is to find a collision free path to take a moveable object from a start configuration to a goal configuration while avoiding obstacles and self-collisions. This problem has application in medicine, robotics, gaming/virtual reality, and search and rescue operations. Exact motion planning methods become intractable as the complexity of the robot increases [18]. Sampling-based motion planning addresses this problem by generating a subset of nodes representing the robot's configuration space, connecting them, and producing a graph containing feasible trajectories.

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), and by 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, S. Jacobs, 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, sjacobs, sthomas, amato}@cse.tamu.edu.

Many methods exist for the various tasks involved in sampling-based motion planning, but selecting the best one for a particular input problem is extremely difficult. This issue is only magnified in heterogeneous environments where different algorithmic choices may apply for different regions. A similar need arises in parallel processing where subdivision is often used to increase parallelism so each region can be processed independently. With different algorithmic choices to be made, there is a need to select appropriate ones automatically and adaptively. For example, Hybrid PRM [6] uses a machine learning approach to dynamically decide which sampling method to use. However, the problem of selecting good candidates for node connection is still daunting and there is no automated way to make this choice.

We introduce Adaptive Neighbor Connection (ANC), a strategy inspired by the same need as Hybrid PRM: different problems and/or regions require different algorithmic choices which are difficult to determine a priori. ANC takes in a list of neighbor finders (NF) and automatically determines the best one to use at a given time. Ideally, ANC should:

- pick a NF that is most likely to successfully connect nodes frequently and punish those that continually do not,
- ensure that all NFs have some chance of being picked,
- consider the cost of rewarding/penalizing them, and
- adapt to changes in performance.

As shown in our results, ANC rapidly learns the best strategy to employ based on a trade-off between success rate and cost. It is able to adapt to changing sampling density as roadmaps are incrementally constructed or to different region types when a problem is partitioned.

We compare ANC to 5 other popular connection strategies over a variety of environments including articulated linkages up to 37 degrees of freedom (DOF). In scenarios where roadmaps are incrementally constructed until they solve the query, ANC is either the best or near the best performer. Over all problems studied, ANC differs the least from hand picking the best. In fixed time scenarios, ANC is able to adapt to the ever changing sampling density. We also show how ANC naturally fits in a parallel setting where the problem is partitioned and a strategy must be learned for each region.

II. Preliminaries and Related Work

Probabilistic Roadmap Methods (PRMs) [11] are sampling-based motion planning methods that comprise a two stage process: roadmap construction and query processing. During roadmap construction, PRMs sample the configuration space (C-Space), retaining valid ones, and

attempt to connect them using some local planner. C-Space comprises all possible placements of the robot, valid or not [13].

Especially relevant to our work is Hybrid PRM which uses more than one sampling strategy and adaptively learns which method to employ over time [6]. We use a similar approach here, but we apply this methodology to the connection phase. Other work has investigated learning from prior execution, particularly for collision checking and local planning [16] where they use historical information from collision calls to compute an approximate C-Space representation via a hash table. They show an improvement in connectivity, but this approach is limited to low DOF problems.

A. Candidate Neighbor Selection Methods

There have been a number of methods proposed for locating candidate neighbors for connection. It is intractable to simply attempt all possible connections since the time to do so is $O(n^2)$. Geraerts and Overmars [5] describe the properties of these neighbor finding approaches and motivate research on connections based upon Reachability Analysis. However, these are expensive and should be limited to acquiring roadmap connectivity and/or seeking asymptotically shortest paths [10].

The most common method for PRMs is the K-Closest approach which returns the k closest neighbors to a node based on some distance metric, where k is normally some small constant, and can be done in logarithmic time. The advantage is that nodes are more likely to be connectable by the local planner because the volume of C-Space the connection occupies is smaller. A similar approach is the r-closest method which returns all neighbors within a radius r of the node as determined by some distance metric. Here, the size of the neighbor set is not fixed but is dependent on the sampling density.

Two randomized variants of these methods are proposed in [14]: 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. In some cases, these methods outperform K-Closest as they introduce some useful randomness.

Other methods use data structures to more efficiently compute nearest neighbors. *Metric Trees* [22] organize the nodes 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 [12]. *KD-trees* [2] 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.

Approximate neighbor finding methods address the running time issue by instead returning a set of approximate K-Closest neighbors. These include spill trees [12], MPNN [23], and Distance-based Projection onto Euclidean Space [17]. These methods usually provide a bound on the approximation error.

B. 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 a PRM predicts how likely it is that a pair of nodes can be connected. In this paper, we study the set of distance metrics commonly used in PRMs:

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. In the results presented here, we use $s = 0.5$ and refer to this as "Euclidean".

Center of Mass: This is the Euclidean distance between the center of mass of the robot at a and at b.

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.

C. Spatial Subdivision and Parallelism

A real world motion planning problem is usually non-uniform and heterogeneous, e.g., a house or factory floor is composed of logically separate areas. Spatial subdivision and region identification improves roadmap quality in these types of environments [3], [15], [20], [24] and naturally lends itself to parallel processing. Previous work has shown that by subdividing the space, scalable performance to large processor counts can be achieved [7], [8], [19].

III. Adaptive Neighbor Connection

The Adaptive Neighbor Connection (ANC) strategy introduced in this paper generates a set of candidate neighbors for a node q for PRM connection using a list of neighbor finders $nf_1, nf_2, \ldots, nf_m$. ANC learns a selection probability for each NF based on its prior success rate and cost.

ANC observes the performance of the local planner on the neighbors returned by the NF. Similarly to how Hybrid PRM [6] selects various samplers during roadmap construction, ANC selects NFs by maintaining a probability p_i for each nf_i.

One way to evaluate nf_i is to determine how many of the neighbors returned resulted in a successful connection. As each NF is used, ANC monitors performance and naturally

favors those with good performance and invokes them more frequently. Algorithm 1 gives a description of ANC during PRM roadmap construction.

Algorithm 1 ANC

Input. A connecting vertex q, a set of neighbor finders NF, a local planner lp and a graph G

Output. A connected graph G with additional edges

Require: Let P be a set of probabilities such that p_i is the probability of selecting nf_i . Initialize $p_i = 1/|NF|$, $\forall p_i \in P$

1: Randomly pick nf_i according to P
2: $N = nf_i$.FIND_NEIGHBORS(q, G)
3: **for** each $n \neq q \in N$ **do**
4: **if** lp.IS_CONNECTABLE(q, n) **then**
5: G.ADD_EDGE(q, n)
6: **end if**
7: **end for**
8: Let r be the success rate of lp over N
9: Let c be the cost incurred
10: Update (P, r, c) according to Equation 3 and Equation 4.

A. Learning Selection Probabilities

ANC learns the best NF to use based on its performance over time. If successful connections increase, the NF gets rewarded and its selection probability is increased. Otherwise, it is punished by decreasing its probability. In addition, if its execution time is expensive, we lower its probability. In the results presented here, we calculate the cost as the number of collision detection calls recorded by the planner. Collision checking takes up a large portion of the computation time for neighbor finding and thus is a good measure of cost.

ANC maintains a weight for each NF similar to Hybrid PRM [6]. These weights keep track of the past performance of the NF. ANC initializes each weight w_i to 1. Based on the weights, ANC computes in a step-wise manner a probability p_i^* for nf_i that is insensitive to the cost:

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

where $w_i(t)$ is the weight of nf_i in step t, t is the number of connection attempts made by the planner, and γ is a fixed constant. The probability p_i^* is a weighted sum of the relative weight of nf_i and the uniform distribution. This ensures that each NF has some chance of being selected.

Let x_i be the reward for the nf_i that was selected. All other rewards for that time step are 0. To update the weights, we first take into account an adjusted reward that is not dependent on the cost accrued (calculated as the cost insensitive probability):

$$x_i^* = x_i/p_i^*, i = 1, 2, ...m. \quad (2)$$

Then we update the weights for all the neighbor finders:

$$w_i(t+1) = w_i(t)\exp\frac{\gamma x_i^*}{m}, i = 1, 2, ...m. \quad (3)$$

The new weight is the current weight multiplied by a factor that depends on the reward received. The exponential factors enable the weights to adapt quickly.

We now include the cost in the selection probability:

$$p_i = \frac{p_i^*/c_i}{\sum_{j=1}^{m} p_j^*/c_j}, i = 1, 2, ...m, \quad (4)$$

where c_i is the average cost of attempting to connect i. Thus, a high cost NF has a smaller selection probability.

B. ANC and Spatial Subdivision and Parallelism

Recall that many problems are well-suited for spatial subdivision, either due to their heterogeneity or to employ parallelism. ANC naturally fits into this framework by adaptively selecting the appropriate connection method for each region. ANC initializes a set of selection probabilities for each region. Learning then proceeds as described in each region independently. In parallel processing scenarios, this puts no additional strain on communication, a critical barrier to scalability.

If the environment is indeed heterogeneous, ANC performance would be hampered if the environment is not partitioned into regions because ANC would be forced to chose some neutral strategy or to vacillate between several strategies. In such a situation, it is desireable to subdivide the problem into homogeneous regions and apply ANC in each one. As it can be hard to know how to subdivide, one option is to over partition the problem to increase the likelihood of homogeneous regions. This naturally lends itself to parallel processing each region independently. In fact, [7], [8], [19] uses spatial decomposition to increase parallelism and is ideally suited for this.

IV. Experiments

We compare ANC to several other popular connection strategies. We first provide details on the experimental setup in Section IV-A. In Section IV-B roadmaps are incrementally constructed until a query is solved for a variety of robots. Section IV-C studies the performance for 21 and 37 DOF articulated linkages with a bounded computation time. Finally, Section IV-D shows the usefulness of ANC with region subdivision needed in heterogeneous environments and in parallel settings.

A. Experimental Setup

We implement ANC in the C++ motion planning library which uses the Standard Template Adaptive Parallel Library (STAPL), a C++ parallel library [4], [21]. We use RAPID [9] for collision detection. Results are averaged over 10 random seeds. For the parallel experiments, we perform a single run and we use a AMD Opteron 2350 processors quad-core, with 8 cores per node, 2.5GHz, 160GB internal disk on each node and 32GB DDR2 800MHz.

We study the following environments (see Figure 1):

- **Maze, spherical rigid robot.** (Figure 1(a)) 6-DOF rigid-body robot that must pass through a series of

tunnels, avoiding some dead-ends, from the top to the bottom. Here, there are two large free areas connected by long narrow passages, and the obstacle occupies the majority of the planning space.

- **U-Tunnel, rod-like rigid robot.** (Figure 1(b)) The robot must navigate through a u-shaped tunnel with two wide passages and a slender passage between them.
- **Cluttered, cube-like rigid robot.** (Figure 1(c)) A box object must pass through different sized passages.
- **Walls, {10, 21, 37} DOF linkage.** (Figure 1(d)) This environment has 10, 21 and 37 DOF free-flying robot scenarios connected by revolute joints. The thin walls and openings produce situations where exact nearest-neighbor configurations will be difficult to connect.
- **2D-Heterogeneous, rod-like rigid robot.** (Figure 1(e)) This environment has 8 different rooms of different types including cluttered, free, and blocked.
- **3D-Heterogeneous, spherical rigid robot.** (Figure 1(f)) This environment has 4 regions separated by walls with single openings. Two regions resemble the Maze environment (Figure 1(a)), one region is comprised of parallel plates producing narrow passages, and one region has randomly placed plates.

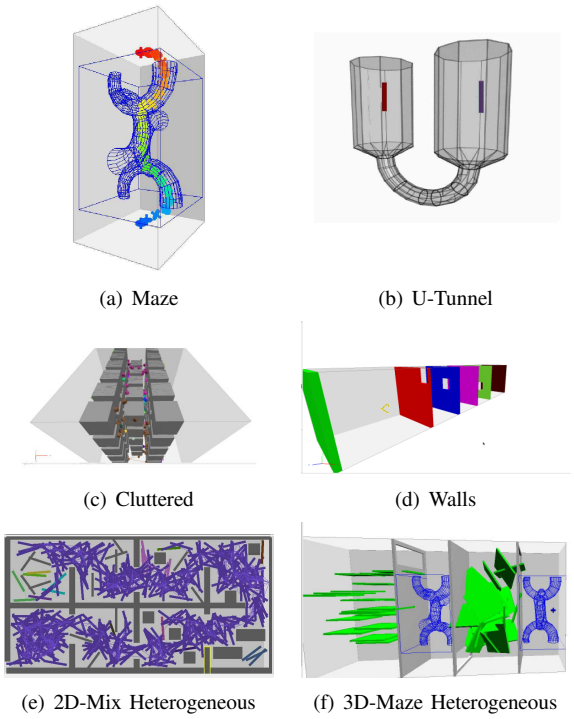
We use obstacle-based sampling [1] for the rigid body problems and uniform random sampling for the linkage problems. Connections are attempted between a node and its neighbors using a straight-line local planner. Neighbors are either defined as the exact k nearest neighbors as given by some distance metric (K-Closest), as k randomly selected neighbors from the exact k_2 nearest neighbors (K-Closest,K-Rand) where $k_2 = 3k$ as in [14], or as k randomly selected neighbors from within a radius r (R-Closest,K-Rand). For the parallel experiments, we use 1, 2, 4, 8, and 16 processors. Distance metrics include scaled Euclidean with $s = 0.5$ (Euclidean), center of mass (COM), and local planner swept volume (lpswept). ANC takes all the above connection strategies as input.

To examine performance, we look at the size of the final roadmap, its average vertex degree (Edge/Nodes), the total time needed, the total number of connected components (CC), and the roadmap connectivity. Roadmap connectivity is defined here as the percentage of nodes in the largest CC. For the parallel experiments, we look at scalability and the ability to produce roadmaps that would solve a given query.

B. Querying the Environment

We analyze the effectiveness of ANC on each of the environments in Figure 1. Table I reports the results for each scenario averaged over 10 runs. Boldface entries indicate the methods that produced the most desirable result for each characteristic (e.g., smallest roadmap size, shortest running time, greatest Edges/Nodes ratio, etc.).

In the Maze environment (Figure 1(a)) with $k = 10$, ANC comes second to K-Closest,K-Rand(Euclidean) and R-Closest,K-Rand(Euclidean) (which records approximately the same time) in terms of time to solve the query and produces a roadmap with the highest average degree second



(a) Maze (b) U-Tunnel (c) Cluttered (d) Walls (e) 2D-Mix Heterogeneous (f) 3D-Maze Heterogeneous

Fig. 1: Problems studied. (a) A spherical rigid body robot in the Maze. The swept volume shows an example trajectory. (b) A long rigid body robot in the U-tunnel. (c) A cube-like rigid body robot must navigate inside different sized narrow passages. A sample roadmap is shown. (d) Articulated linkages (10, 21, and 37 DOF) must travel through a narrow hole in each wall. (e) A long rigid body robot in a heterogeneous 2D environment. A sample roadmap is shown. (f) A spherical rigid body robot in a heterogeneous 3D environment.

to the connector method that it learnt as shown in Figure 2. Such a roadmap is useful when path quality is a concern (e.g., paths with high clearances or low power consumption). Note that while ANC is not the fastest in this environment, three of the other methods are much slower, some by several orders of magnitude. Thus, different connection methods have significantly different levels of success here and selecting one or more of them to use intelligently is critical. We also study $k = 20$. The results show the same trend as the previous experiment with $k = 10$. Thus, ANC is invariant to k in terms of performance.

Figure 2 plots the probability that each connection method is selected in the ANC framework over time for a single representative run. Figure 2 shows that early on R-Closest,K-Rand(Euclidean) is selected. However, as roadmap construction progresses, the success of this method begins to level out and the probability of K-Closest,R-Rand(Euclidean) begins to increase and then levels off. This behavior in the learning plot is indicative of the performance of these two connection methods in Table I. It also shows ANC's ability in certain

TABLE I: Each method constructs a roadmap until the query is solved. ANC is comprised of the other 5 connection methods. All results are averaged over 10 runs. CC is number of connected components present. CC_i% = percentage of nodes in the ith component. Boldface entries indicate the most desirable (e.g., shortest running time, greatest Edges/Nodes ratio).

Environment	Method	Nodes	Edges/Nodes	Total Time (s)	CC	CC_1%	CC_2%	CC_3%
Maze, spherical rigid body robot, $k = 10$	ANC	960	13.92	218	97	85.6	0.3	0.2
	K-Closest(Euclidean)	1483	13.11	413	209	84.7	0.7	0.3
	K-Closest,K-Rand(Euclidean)	**502**	13.60	**100**	57	88.4	0.2	0.2
	R-Closest,K-Rand(Euclidean)	502	**14.60**	101	**16**	**96.6**	0.5	0.1
	K-Closest(COM)	989	13.30	1626	158	83.2	0.4	0.3
	K-Closest(lpswept)	1002	13.67	1041	42	89.8	0.6	0.5
Maze, spherical rigid body robot, $k = 20$	ANC	496	14.31	1312	93	64	13	2
	K-Closest(Euclidean)	496	14.12	1305	104	63	12	2
	K-Closest,K-Rand(Euclidean)	**496**	13.43	**1294**	123	52	10	6
	R-Closest,K-Rand(Euclidean)	496	**14.79**	1320	72	88	0.2	0.2
	K-Closest(COM)	496	14.06	1693	**101**	**62**	12	2
	K-Closest(lpswept)	496	14.21	2086	85	71	3	0.2
Cluttered, cube-like rigid body robot, $k = 10$	ANC	**537**	10.274	974	29.57	93.66	0.72	0.48
	K-Closest(Euclidean)	**537**	10.326	**861**	17.29	93.78	3.07	0.31
	K-Closest,K-Rand(Euclidean)	573	**11.169**	1330	25.64	**95.06**	0.50	0.26
	R-Closest,K-Rand(Euclidean)	**537**	10.328	890	**17.01**	93.81	2.98	0.21
	K-Closest(COM)	**537**	10.332	977	17.43	93.77	3.07	0.31
	K-Closest(lpswept)	**537**	10.321	991	17.43	93.77	3.07	0.31
U-Tunnel, rod-like rigid body robot, $k = 10$	ANC	**14164**	15.164	**308**	6.4	66.68	7.03	5.20
	K-Closest(Euclidean)	21851	14.809	761	5.4	66.99	7.09	5.28
	K-Closest,K-Rand(Euclidean)	30323	12.048	610	5.3	66.99	7.09	5.19
	R-Closest,K-Rand(Euclidean)	21602	**15.839**	768	**4.2**	67.03	7.09	5.38
	K-Closest(COM)	31108	15.107	927	7.4	66.38	5.28	4.66
	K-Closest(lpswept)	17444	14.621	5780	7.2	**72.02**	0.08	0.08
Walls, 10 DOF free-flying articulated linkage, $k = 10$	ANC	**644**	**18.594**	26	3.21	45.87	23.62	16.08
	K-Closest(Euclidean)	1465	11.799	**22**	4.64	44.71	30.08	10.81
	K-Closest,K-Rand(Euclidean)	3357	11.325	60	8.14	65.39	31.98	1.27
	R-Closest,K-Rand(Euclidean)	1322	17.257	37	**2.57**	**82.60**	3.07	0.05
	K-Closest(COM)	1215	10.505	107	8.07	34.48	32.92	10.81
	K-Closest(lpswept)	1358	10.586	125	7.57	40.50	32.99	11.92

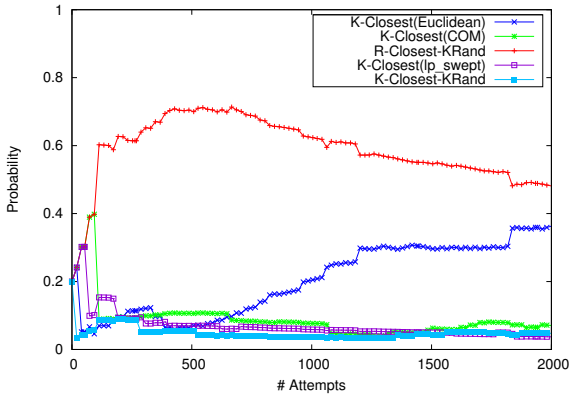


Fig. 2: Learning plot for the maze environment with $k = 10$.

scenarios to utilize the performance of the better connection method in the set if one exists. Thus, it is important that ANC adapts continuously over the roadmap construction process as the best method to employ changes over time. This pattern of early learning versus later learning was also seen in Hybrid PRM with sampler selection [6].

In the Cluttered environment, K-Closest,K-Rand (Euclidean), which was the best performer in terms of running time for the Maze, is the worst performer. Clearly, one cannot select the same connection method for different environments and expect similar performance, even when both robots are similar as in this case. Instead, K-Closest (Euclidean) emerges as the winner in running time. However, note that no one method performs optimally across all the metrics.

The U-tunnel environment has a long rigid body robot which behaves much differently than the compact robots in the prior environments. Here, ANC performs significantly better than the other methods in terms of running time and roadmap size and near the best in terms of Edges/Nodes ratio. Figure 3 provides the learning plot. Again, ANC is able to adapt and learn a good connection strategy.

For the 10 DOF articulated linkage in the Walls environment, ANC comes second fastest in running time but best in roadmap size and Edges/Nodes ratio. ANC starts by learning K-Closest,K-Rand (Euclidean), see Figure 4. As time goes on, it increases the probability of K-Closest (Euclidean) and then the other methods after. ANC is able to dynamically adjust probabilities based on the performance of each method.

We can see from each of these environments that no connection strategy is always the best. In fact, picking a single connection strategy for a heterogeneous environment

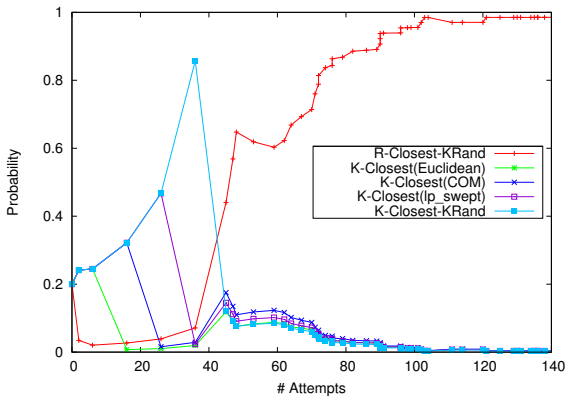


Fig. 3: Learning plot for the U-tunnel environment.

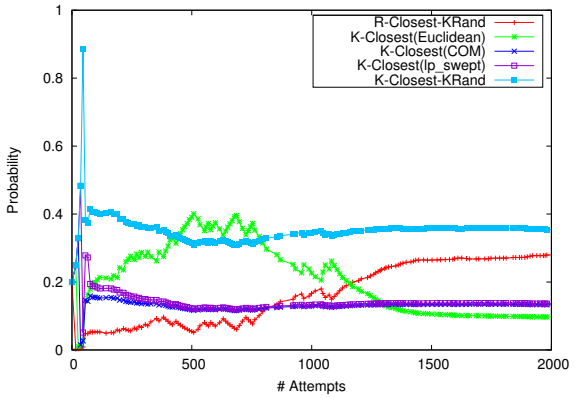


Fig. 4: Learning plot for a 10 DOF linkage in the Walls environment.

can be quite detrimental. Table II shows the percentage difference between each strategy and the best strategy in terms of running time for all of the environments. ANC consistently accrues the least percentage difference across all environments and is only a fraction of some of the others.

C. Connectivity Analysis within a Specified Time Range

We perform experiments using the 21 DOF and 37 DOF articulated linkages in fixed time scenarios. Results are averaged over 10 random seeds. We fix the time for the 21 DOF experiments to 750 seconds and the 37 DOF to 1500 seconds.

In Table III we see that there is no clear winner for this environment. For both robots, ANC is able to learn the best connectivity method and performs second best, which is to be expected as shown in Figure 5. Figure 5 shows that as the sampling density increases (over time), R-Closest,K-Rand (Euclidean) increases in probability indicating how ANC adapts. Statically picking one method cannot achieve this.

D. ANC with Spatial Subdivision and Parallelism

We perform experiments on two different heterogeneous environments: a 2D environment with a rod-like robot and a 3D environment with a spherical robot, see descriptions in Section IV-A. We start by generating 100 nodes in each

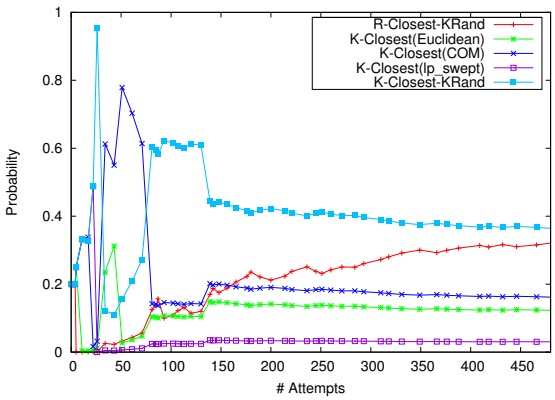


Fig. 5: Learning plot for the 37 DOF linkage in the Walls environment.

region and attempt to solve a query. We continue adding nodes and edges until the query is solved. Both environments are decomposed into 8 regions on a grid. Regions are numbered starting from the top left and ending on the bottom right, i.e., the top row is numbered 1, 2, 3, 4 and the bottom row is numbered 5, 6, 7, 8 from left to right. Each method generates roadmaps in the different regions and then attempts additional connections between regions to form one roadmap.

Table IV shows that ANC indeed learns different connectors in the 4 different regions (out of 8) shown. K-Closest (lpswept), though expensive, is beneficial in improving connectivity for robots with a large swept volume as for the rod-like robot here. Thus, it is learned in region 8 where the room contains two long obstacles with a narrow passage where rotations would likely be invalid. Yet in most other regions, this connector is given a low probability as other, cheaper connectors can make sufficient connections.

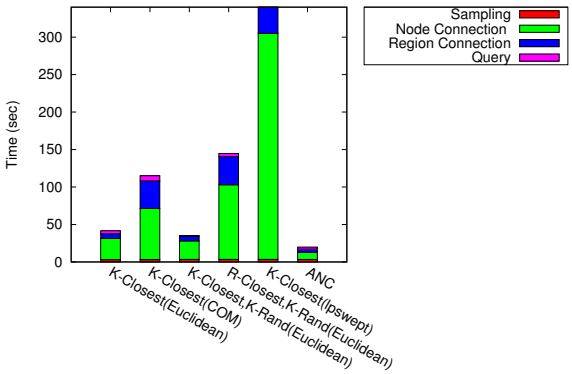


Fig. 6: Running time for each method in the 2D-Heterogeneous environment.

Figure 6 shows the total running time for each method broken down into four main phases: sampling, connection within regions, connection between regions, querying. ANC outperforms all the other methods as it efficiently selects appropriate connection methods to use.

Table V shows the same ability of ANC to learn different

TABLE II: Comparison of the % difference in running time against the best strategy across all environments.

Method	Maze (%)	Cluttered (%)	U-Tunnel (%)	Walls, 10 DOF (%)	**Total (%)**
ANC	118.4	13.1	0.0	18.2	**149.7**
K-Closest(Euclidean)	313.4	0.0	147.1	0.0	**460.5**
K-Closest,K-Rand(Euclidean)	0.0	54.5	98.1	172.7	**325.3**
R-Closest,K-Rand(Euclidean)	1.6	3.4	149.4	68.2	**222.6**
K-Closest(COM)	1516.8	13.5	201.0	386.4	**2117.7**
K-Closest(lpswept)	914.3	15.1	1776.6	468.2	**3174.2**

TABLE III: Each method constructs a roadmap for a fixed amount of time, 750 seconds for the 21 DOF linkage and 1500 seconds for the 37 DOF linkage, in the Walls environment. ANC is comprised of the other 5 connection methods. All results are averaged over 10 runs. CC is number of connected components present. CC_i% = percentage of nodes in the ith component. Boldface entries indicate the most desirable (e.g., shortest running time, greatest Edges/Nodes ratio).

Environment	Method	Nodes	Edges/Nodes	CC	CC_1%	CC_2%	CC_3%
Walls, 21 DOF free-flying articulated linkage $k = 10$	ANC	**87.50**	9.17	4.21	36.35	21.80	20.13
	K-Closest(Euclidean)	64.86	9.45	4.14	35.28	27.56	16.90
	K-Closest(COM)	64.21	9.54	4.07	36.77	27.74	16.29
	K-Closest,K-Rand(Euclidean)	82.43	9.14	4.36	**41.11**	28.54	11.48
	R-Closest,K-Rand(Euclidean)	62.00	**10.31**	**3.93**	37.39	28.91	16.32
	K-Closest(lpswept)	64.00	9.50	4.07	36.81	27.83	16.15
Walls, 37 DOF free-flying articulated linkage $k = 10$	ANC	49.50	7.461	**4.29**	33.51	22.29	20.24
	K-Closest(Euclidean)	50.71	7.572	4.71	32.82	21.51	18.94
	K-Closest(COM)	50.71	7.570	4.64	31.71	21.63	20.56
	K-Closest,K-Rand(Euclidean)	**53.14**	7.210	4.79	**33.94**	24.78	18.53
	R-Closest,K-Rand(Euclidean)	52.86	7.574	5.64	25.69	22.11	17.06
	K-Closest(lpswept)	50.71	**7.579**	6.45	31.40	20.43	19.56

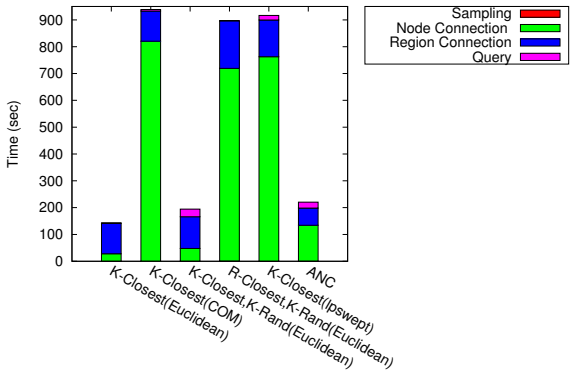


Fig. 7: Running time for each method in the 3D-Heterogeneous environment)

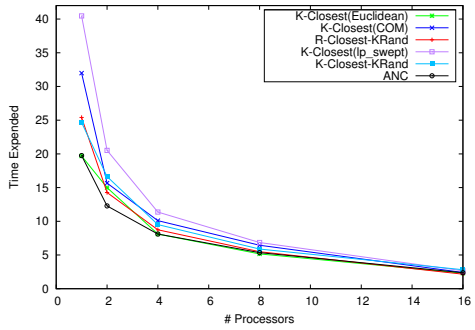


Fig. 8: Running times for various methods in the 2D-Heterogeneous environment with up to 16 processors.

connection methods for the 3D-Heterogeneous environment. Figure 7 shows that although ANC is not the best performing method here, 3 out of 5 of the other connection methods take orders of magnitude longer than ANC. Here we see that ANC makes the best of poor connection method choices available.

To demonstrate how ANC works in a parallel setting, we subdivide the 2D-Heterogeneous environment into 32 regions and construct roadmaps of equal size. Figure 8 shows that ANC is one of the fastest methods and scales well. Thus, in a parallel setting with region subdivision, ANC does not hamper performance.

V. Conclusion

This paper looks into an intelligent way of choosing connection strategies for generating PRM roadmaps. It uses a reward and cost approach to determine how well a connection strategy performs and decides if it is viable in that particular environment. We study a host of experiments to confirm that the different connection strategies perform differently in varying environments, and we show that our method relieves the burden of deciding which connection strategy to employ. Our method is able to adapt to the environment and is very useful in different environment types. We are also able to adapt to changing sample density, which other methods are unable to achieve. Our results show we are able to get connectivity that is on par with the best connection strategy,

TABLE IV: Final selection probabilities for connectors in the 2D-Heterogeneous environment for several representative regions. Boldface entries indicate the winning probability in each. ANC is able to learn different probabilities for each type.

Region Number	Region Type	Final Selection Probability				
		K-Closest (Euclidean)	K-Closest (COM)	K-Closest,K-Rand (Euclidean)	R-Closest,K-Rand (Euclidean)	K-Closest (lpswept)
1	stick obstacles	0.073	0.084	**0.704**	0.074	0.062
4	2 boxes	0.054	0.058	**0.752**	0.089	0.048
5	free environment	0.011	0.129	0.128	**0.648**	0.082
8	2 long rods	0.262	0.081	0.115	0.126	**0.419**

TABLE V: Final selection probabilities for connectors in the 3D-Heterogeneous environment for several representative regions. Boldface entries indicate the winning probability in each. ANC is able to learn different probabilities for each type.

Region Number	Region Type	Final Selection Probability				
		K-Closest (Euclidean)	K-Closest (COM)	K-Closest,K-Rand (Euclidean)	R-Closest,K-Rand (Euclidean)	K-Closest (lpswept)
2	Maze	0.003	0.001	0.006	**0.906**	0.081
3	Planes	0.084	0.104	0.323	0.094	**0.392**
4	Maze	0.033	0.023	0.029	**0.596**	0.319
8	Planes	**0.232**	0.185	0.183	0.214	0.184

and in some cases outperforms it. Our method has the lowest time overhead overall in all the environments studied. We also use parallel computation and decomposition to further show the usefulness of our approach.

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 on Robotics : the algorithmic perspective: the algorithmic perspective*, WAFR '98, pages 155–168, Natick, MA, USA, 1998. A. K. Peters, Ltd.

[2] 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.

[3] R. A. Brooks and T. Lozano-Pérez. A subdivision algorithm in configuration space for findpath with rotation. In *Proc. Int. Conf. Artif. Intel.*, pages 799–806, 1983.

[4] A. Buss, Harshvardhan, I. Papadopoulos, O. Pearce, T. Smith, G. Tanase, N. Thomas, X. Xu, M. Bianco, N. M. Amato, and L. Rauchwerger. STAPL: Standard template adaptive parallel library. In *Proc. Annual Haifa Experimental Systems Conference (SYSTOR)*, pages 1–10, New York, NY, USA, 2010. ACM.

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

[6] 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.

[7] S. A. Jacobs, K. Manavi, J. Burgos, J. Denny, S. Thomas, and N. M. Amato. A scalable method for parallelizing sampling-based motion planning algorithms. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2012.

[8] S. A. Jacobs, N. Stradford, C. Rodriguez, S. Thomas, and N. M. Amato. A scalable distributed rrt for motion planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2013.

[9] D. Johnson and E. Cohen. A framework for efficient minimum distance computations. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 4, pages 678–3684, 1998.

[10] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research (IJRR)*, 30:846–894, 2011.

[11] 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.

[12] 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.

[13] T. Lozano-Pérez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, October 1979.

[14] 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.

[15] M. A. Morales A., L. Tapia, R. Pearce, S. Rodriguez, and N. M. Amato. C-space subdivision and integration in feature-sensitive motion planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3114–3119, April 2005.

[16] 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.

[17] 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.

[18] J. H. Reif. Complexity of the mover's problem and generalizations. In *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, pages 421–427, San Juan, Puerto Rico, October 1979.

[19] C. Rodriguez, J. Denny, S. A. Jacobs, S. Thomas, and N. M. Amato. Blind rrt: A probabilistically complete distributed rrt. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2013.

[20] 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.

[21] G. Tanase, A. Buss, A. Fidel, Harshvardhan, I. Papadopoulos, O. Pearce, T. Smith, N. Thomas, X. Xu, N. Mourad, J. Vu, M. Bianco, N. M. Amato, and L. Rauchwerger. The STAPL Parallel Container Framework. In *Proc. ACM SIGPLAN Symp. Prin. Prac. Par. Prog. (PPoPP)*, pages 235–246, San Antonio, Texas, USA, 2011.

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

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

[24] L. Zhang, Y. Kim, and D. Manocha. A hybrid approach for complete motion planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 7–14, 2007.