10-2020

Topology-Guided Roadmap Construction With Dynamic Region Sampling

Read Sandström

Diane Uwacu

Jory Denny
University of Richmond, jdenny@richmond.edu

Nancy M. Amato

Recommended Citation

Sandstrom, Read, Diane Uwacu, Jory Denny, and Nancy M. Amato. "Topology-Guided Roadmap Construction With Dynamic Region Sampling." *IEEE Robotics and Automation Letters* 5, no. 4 (October 2020): 6161–68. https://doi.org/10.1109/LRA.2020.3010487.

Topology-Guided Roadmap Construction with Dynamic Region Sampling

Read Sandström[1] Diane Uwacu [1] Jory Denny[2] Nancy M. Amato[3]

***Abstract*—Many types of planning problems require discovery of multiple pathways through the environment, such as multi-robot coordination or protein ligand binding. The Probabilistic Roadmap algorithm (PRM) is a powerful tool for this case, but often cannot efficiently connect the roadmap in the presence of narrow passages. In this paper, we present a guidance mechanism that encourages the rapid construction of well-connected roadmaps with PRM methods. We leverage a topological skeleton of the workspace to track the algorithm's progress in both covering and connecting distinct neighborhoods, and employ this information to focus computation on the uncovered and unconnected regions. We demonstrate how this guidance improves PRM's efficiency in building a roadmap that can answer multiple queries in both robotics and protein ligand binding applications.**

***Index Terms*—Motion and Path Planning, Semantic Scene Understanding**

I. Introduction

PLANNING motions is a fundamental component of many applications including physical robot operations within buildings [1] or biological simulations of ligand binding [2]. In these domains, Probabilistic Roadmap (PRM) [3] approaches are attractive not only because of their ability to encode multiple differing pathways between start and goal locations, but also due to the inherent ability to solve for any number of queries at time. Ultimately this approach can lead to added benefits. For example, in robotics this can lead to robust long-term autonomy [4], while in biological simulations faster and more accurate convergence to a solution [5].

Despite the advantages to the PRM approach, their efficiency quickly degrades as constraints are added to a system, e.g., the need for a robot to navigate corridors with low clearance or a ligand is in close proximity to its binding site on a protein. The degradation in planning performance in these scenarios is well documented and often referred to as the *narrow passage* problem [6].

Manuscript received: Feb., 24, 2020; Revised May, 23, 2020; Accepted June, 26, 2020.

This paper was recommended for publication by Editor Dezhen Song upon evaluation of the Associate Editor and Reviewers' comments. This research supported in part by NSF awards CSF-1439145, CCF-1423111, EFRI-1240483, RI-1217991.

[1]Read Sandström and Diane Uwacu are with the Parasol Laboratory, Dept. of Computer Science and Engineering, Texas A&M University, College Station, TX, 77840, USA. readamus@tamu.edu, duwacu@tamu.edu

[2]Jory Denny is with SpiRoL: Spider Robotics Lab, Dept. of Mathematics and Computer Science, University of Richmond, Richmond, VA, 23173, USA. jdenny@richmond.edu

[3]Nancy M. Amato is with the Parasol Laboratory, Dept. of Computer Science, University of Illinois Urbana-Champaign, Champaign, IL, 61820, USA. namato@illinois.edu

Digital Object Identifier (DOI): see top of this page.

Many potential solutions exist to solving the narrow passage problem, but they often reason about localized information during the planning process. For example, some approaches attempt to build solution paths closer to obstacles or apply denser sampling in difficult planning areas [7], [8]. Despite this, there is not a cohesive approach to constructing PRMs that efficiently promotes both coverage and connectivity.

We propose an approach that balances the benefits of localized reasoning with tactical global exploration in order to provide an efficient planning technique for constructing PRMs. Our approach relies heavily on a connection to a minimal representation of the topology of the workspace, called a workspace skeleton. Essentially, this workspace guide provides direct insight into the relevant portions of the planning space that the roadmap currently covers and does not cover. Over time, sampling is biased towards the frontier of this structure which is succinctly encoded as workspace regions. Thus, the approach exploits workspace regions to focus sampling in a way that achieves quick coverage of a planning space. Our contributions are as follows, we:

- present a novel algorithm that effectively exploits workspace topology for constructing probabilistic roadmaps,
- analyze various non-trivial implementation considerations that affect performance of topologically-inspired approaches, and
- experimentally show that our approach is more efficient at constructing PRMs in a wide variety of scenarios.

In prior work [9], we have shown that a similar topological guidance benefits the planning process for other sampling-based motion planning paradigms, and in this work we extend this methodology in a non-trivial way so that it can be applied to PRM approaches. This approach is targeted at problems where the workspace is closely tied to the planning space, and takes that relationship as an inherent assumption. In our experience, this holds true in many applications including our motivating examples. This work represents a portion of the author's PhD dissertation [10].

II. Related Work

In this paper, we discuss motion planning in the context of holonomic robots, i.e., robots whose *degrees of freedom* (DOFs) contain no velocity constraints. The DOFs for a robot parameterize its pose in the 2-d or 3-d *workspace*. They include, for example, object position, orientation, joint angles, etc. A configuration is a single specification of the DOFs $q = \langle x_1, x_2, \ldots, x_n \rangle$, where x_i is the ith DOF and n is the total number of DOFs. The set of all possible parameterizations is called the *configuration space* ($\mathcal{C}_{space}$) [11]. $\mathcal{C}_{space}$ is often partitioned into two subsets, *free space* ($\mathcal{C}_{free}$) and *obstacle space* ($\mathcal{C}_{obst}$). Given a start configuration and a goal configuration or region, the motion planning problem is the problem of finding a continuous trajectory in $\mathcal{C}_{free}$ between the start and goal. We define a *query* as a start and goal pair.

In general, it is infeasible to explicitly compute a representation of $\mathcal{C}_{obst}$ [12], but we can often determine the validity of a configuration q efficiently with a workspace collision test between the robot placed at q and the environment. If the robot placed at q does not collide with itself or the environment, then $q \in \mathcal{C}_{free}$ and is said to be *valid*.

Regions. We define a *region* as any bounded volume in the workspace, e.g., axis-aligned bounding boxes (AABBs) and bounding spheres (BSs). Each point p of a region R maps to a possibly infinite number of configurations in $\mathcal{C}_{space}$, e.g., by placing the center of mass of the robot at p and randomizing the remaining DOFs. The regions will share dimension with the workspace (i.e. spheres for 3D or circles for 2D).

Homotopy. Two paths are defined to be *homotopy equivalent* if and only if one path can be continuously deformed to the other without transitioning through an obstacle region. A *homotopy class* is a set of paths such that any two are homotopy equivalent.

A. Sampling-based Planning

Due to the high complexity of motion planning [12], research methodologies tended toward randomized, sampling-based approaches which attempt to construct a graph, called a *roadmap*, that is an approximate model of $\mathcal{C}_{free}$. While there are many general sampling-based paradigms in the realm of motion planning, we focus our study on Probabilistic RoadMaps (PRMs) [3].

Generally, PRMs iterate between sampling configurations from $\mathcal{C}_{free}$ and connecting nearby configurations together to form the pathways encoded by a roadmap. Due to the randomized sampling, the performance of PRMs degrade as the problem becomes less expansive [6], commonly referred to as the *narrow passage problem*.

There have been many approaches to addressing the narrow passage problem, both in terms of altering the sampling process and the connection process [7], [8], [13]–[15] (to cite a few). Generally, these fall into two categories: those attempting to plan close to obstacles, e.g., OBPRM [7], and those planning away from obstacles, e.g., MAPRM [14]. However, these approaches altogether use heuristic localized reasoning to improve planning. There is no globalized exploration strategy by which overly sampling one portion of the space is reduced.

1) Workspace-biased Planners: One class of planners use workspace information to aid in the planning process, as a partial step to allowing a global view to constructing a PRM. Here we describe a few.

Feature Sensitive Motion Planning [16] recursively subdivides the space into "homogeneous" regions (regions of the environment containing similar properties, e.g., free or clutter), individually constructs a roadmap in each region, and merges them together to solve the aggregate problem. This framework adaptively decides the specific planning mechanism to map to each homogeneous region.

Other approaches utilize workspace decompositions to find narrow or difficult areas of the workspace to bias $\mathcal{C}_{space}$ sampling [17]–[20]. These methods begin by decomposing the workspace using an adaptive cell decomposition [18] or a tetrahedralization [17], [19], and then they weight the decomposition to bias sampling. However, static determination of sampling regions often leads to oversampling in fully covered regions. Workspace Connectivity Oracle [19] mitigates this by preferring regions that bridge separate connected components.

SyCLoP [20] employs a graph-search over the cell decomposition to lead a search and samples regions near the frontier of the resulting cell path. While similar in spirit to the method presented in this paper, it is applicable only to rapidly-exploring tree (RRT) approaches.

One planning approach proposed allowing a user to define and manipulate regions of the workspace to bias probabilistic roadmap construction [21]. In this work, we utilize a similar concept of workspace regions, but do not rely on a human operator to direct region manipulation.

Despite these advances, no approach cohesively combines and balances a local exploitation strategy with a globalized reasoning for efficiency.

B. Dynamic Region-biased RRT

The predecessor of this work is Dynamic Region-biased RRT [9], which employs a skeleton of the free workspace as a guide for RRT growth. The core idea is to focus sampling in *regions* or volumes of workspace that move along the skeleton edges just ahead of the roadmap frontier. Sampling growth targets within these regions directs the RRT to expand along the paths through the workspace that are defined by the skeleton, leading to fast feasibility planning through intricate workspaces. We refer to this strategy for biasing sampling as *dynamic region sampling*.

Dynamic Region sampling aids planning in narrow passages by focusing sampling in locations the roadmap may need to cover to generate a solution. This significantly reduces the subset of $\mathcal{C}_{space}$ that the sampler must search to discover important narrow passages and thereby expedites their discovery.

As an RRT method, Dynamic Region-biased RRT does not produce highly connected roadmaps. Each vertex has a single path to the root, and paths lying in disjoint regions of $\mathcal{C}_{free}$ will not be discovered at all. To tackle problems requiring good coverage and connectivity of $\mathcal{C}_{free}$ such as multi-query planning, we generalize the dynamic region sampling technique to account for the separate evolution of multiple connected components in the roadmap.

III. METHOD

The method begins by constructing a skeleton of the environment, termed a workspace skeleton. This skeleton is deformation retract of the free workspace, i.e., each point in workspace can be smoothly collapsed to the skeleton in a continuous way [22]. Some examples include the medial

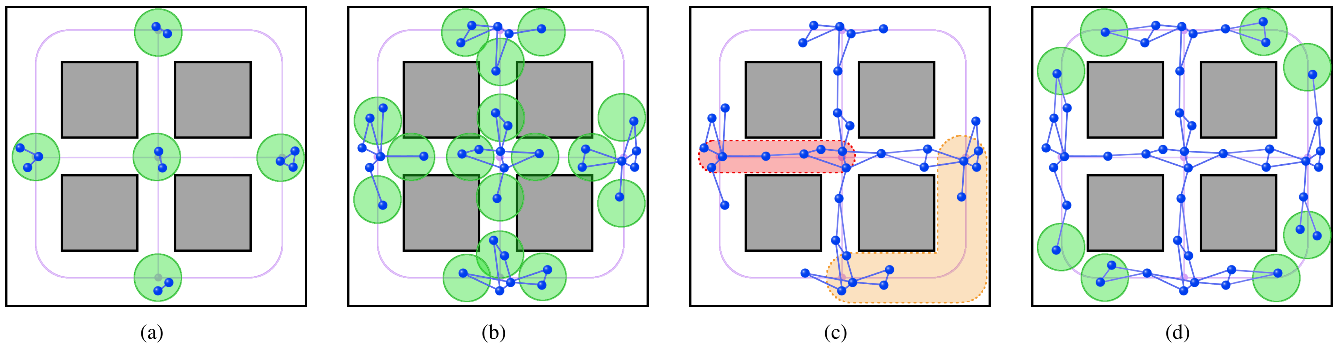


Fig. 1. An illustration of Dynamic Region sampling with PRM. Obstacles are shown in gray. The *workspace skeleton* is shown in purple. (a) The algorithm samples initial connected components (blue) in regions (green) around each skeleton vertex. (b) Sampling regions expand outward along the skeleton edges. We depict the regions in the location where samples were generated for clarity; in the actual algorithm the regions advance past the newly generated samples. (c) An illustration of two edge segments. The red-shaded segment has a single local component which is also a *bridge*. The orange-shaded segment has two distinct local components. (d) The components in the middle tunnels successfully connect to form bridges, and their regions are released. The outer passages are still expanding.

axis skeleton [23] in 2D environments and the mean curvature skeleton [24] in 3D environments.

We use the PRM method as the basis for presentation, but the concepts can be employed with any sampling-based planner that can be cast in terms of expanding and connecting components of the roadmap. The high-level concept for PRM with dynamic region sampling is to view the workspace skeleton as a rough map of the important regions of $\mathcal{C}_{free}$ that we must cover. The skeleton edges describe simple contiguous volumes such as rooms or tunnels, while the skeleton vertices describe junctions of such volumes. We will refer to the volume of workspace described by a skeleton edge as an *edge segment*, which reflects the concept of a skeleton-induced segmentation of the workspace [25].

Our goal will be to cover each edge segment with a set of vertices that connect the roadmap from the region near the source vertex to the region near the target vertex. We describe a roadmap with this property as *locally connected*. Such a roadmap has good coverage of all distinct regions of workspace, and should be able to quickly answer a wide variety of queries with a path of reasonable cost.

To aid in describing the method, we define a *local connected component* for an edge segment as a set of roadmap vertices which are mutually connected without considering vertices outside of the segment volume (Fig. 1(c)). For any two vertices v_a, v_b in a local connected component C, there must be a path from v_a to v_b through some set of vertices $V \subseteq C$. This concept describes a portion of a roadmap that is locally connected within a particular volume of workspace. A local connected component with vertices near both the source and target of the corresponding skeleton edge will be termed a *bridge*. Bridges represent a connected path that traverses the edge segment volume.

The key idea of the method is to generate local connected components near skeleton vertices and extend them across their edge segments with dynamic sampling regions. Local connected components form bridges by either extending all the way across their edge segment or by merging with a local component inbound from the opposite direction.

Initialization: We begin by initializing sampling regions at each skeleton vertex $v \in S_V$ and sampling a number of configurations within (Alg. 1). Next, we attempt to form connections within each group of samples to form one or more connected components at each skeleton vertex. For each such component C, we initialize a sampling region on each outbound edge e from v and track each tuple (C, e.source, e.target) as local connected components. This seeds the roadmap with at least a pair of local connected components for each edge e, with an equal number rooted on either end (Fig. 1(a)). Note that vertices sampled near a skeleton vertex will be present in more than one local component because they are partially responsible for covering each adjacent skeleton edge segment. We initialize sampling regions for each local connected component on the first point in its edge segment to lead extension through the appropriate edge segment.

Expansion: The sampling regions guide expansion of the local connected components they lead. On each iteration of the algorithm, we select a sampling region r and generate one or more configurations Q within its boundary. We then attempt to connect each valid configuration $q \in Q$ to its nearest neighbors in the local connected component C that r is expanding: on failure, q is discarded. Successful connections are retained and added to C (Fig. 1(b), Alg. 2). We then advance r along its skeleton edge path until it no longer touches any of the newly added samples (Alg. 3). In this way, the sampling region r tracks the component C's progress in covering the edge segment.

If r successfully expands C, we additionally attempt to connect the retained samples in Q to any local components inbound on this edge segment from the opposite direction. This is to make the algorithm aggressively attempt to form bridges at the earliest opportunity. On forming a bridge, we merge the newly connected local components and release their sampling regions, which are no longer needed (Fig. 1(d)).

If r advances to the end of its edge without C connecting to a local component rooted at the target skeleton vertex s_t, then C has formed a bridge but not yet connected to the roadmap locally near s_t. In this case, we generate local components with the new vertices Q on the edges outbound from s_t to continue searching for a connection to the local components already rooted at s_t (Fig. 2(b), 2(c)). This ensures that the algorithm continues to explore until the roadmap is locally connected or disjoint global connected components cover the skeleton. The latter can happen in problems with disjoint regions of $\mathcal{C}_{free}$.

Connection: To ensure that disjoint local components within an edge segment are connected, we apply an additional connection stage after each expansion step. We pick a random point p along r's skeleton edge path and sample a set of valid configurations Q. For each $q \in Q$, we attempt to form connections between at least two local connected components within this edge segment. Any configurations that form the necessary connections will be added to the roadmap and will trigger a merge of the corresponding local connected components (Alg. 2).

As in the expansion step, a merge of two components coming from opposite directions forms a bridge and releases their sampling regions. Similarly, a merge with an existing bridge absorbs all affected vertices into the bridge. When two components from the same side merge, we retain the sampling region which has advanced the farthest along the edge path.

Biased Region Selection: The set of initial regions will expand their corresponding components outward from their root skeleton vertex in a similar fashion as in Dynamic Region-biased RRT possibly extending to the end of the skeleton. When selecting a region to expand on each iteration, we can employ a weighted random choice to favor regions which have been more successful in expanding the roadmap. The weight for each region is initialized to one and updated by the success rate of extending into samples generated in that region. To ensure that region weights represent the recent history of performance, we can apply a discount factor $\epsilon \in [0, 1]$ to the prior weight before updates:

$$w \leftarrow \epsilon w + s$$

where s is one if the roadmap connected to the sample and zero otherwise. A weighting based on success rate ensures that the algorithm will explore the edge segments which have proved to be traversable before expending effort on segments which are difficult to connect or even not path-connected in $\mathcal{C}_{free}$.

A. Local Connectivity

A straight-forward application of the dynamic region sampling paradigm for RRT methods is very likely to produce disconnected roadmaps because there is no mechanism ensuring that the samples produced within a region r will connect to other vertices within r's edge segment. This is implied in RRT methods due to tree extension, but not guaranteed for PRM methods which form local plans rather than growing towards a new sample. This motivates our choice of requiring sampling regions to expand a particular local connected component.

Even with this consideration, it is quite possible that local components may grow past each other along a skeleton edge and fail to connect when a connection is feasible (Fig. 2(a)). This motivates the need for a separate connection step to provide a guidance mechanism for completing the connections to achieve local connectivity.

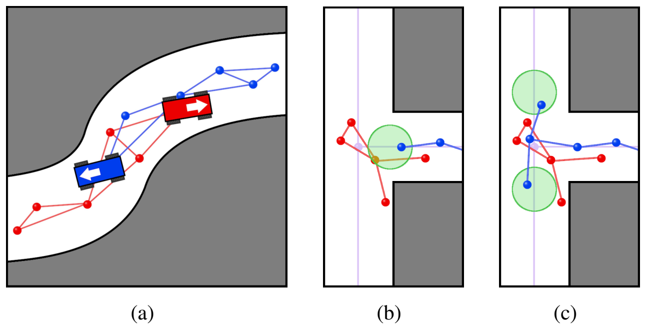


Fig. 2. (a) An example of a missed connection. The robot is a car-like vehicle with mechanum wheels, with an arrow indicating its orientation. Two connected components of the roadmap are shown in red and blue. The components are connectable because the mechanum wheels permit the robot to turn in place. (b-c) An example of a missed bridge. Two local components (red, blue) fail to merge as the blue component approaches the end of its skeleton edge. The blue component forms a bridge on its own, and new regions continue outward from the previous target vertex.

However, we note that a locally connected roadmap does not necessarily express a complete coverage of $\mathcal{C}_{free}$ because $\mathcal{C}_{free}$ can have locally disjoint components within a particular edge segment. Consider an example where a car-like robot must traverse a tunnel that is too narrow to turn around. Within the tunnel, there are two disjoint regions of $\mathcal{C}_{free}$: one for each direction of travel. In more complex examples with three-dimensional environments or mobile manipulators, there could be many more locally disjoint components that are only connected in some specific areas of the environment. The algorithm attempts to account for this by creating new local components when a sampling region completes a skeleton edge without connecting to the other side. This encourages construction of a roadmap that presents some level of coverage for locally disjoint regions of $\mathcal{C}_{free}$.

B. Sampling Regions and Clearance Awareness

We assume topological skeleton of the workspace $S = (S_V, S_E)$ as an input element. Throughout the algorithm, we define sampling regions at vertices and points on the edge paths. These regions may be given some fixed size as in [9], but we observe that an alternate strategy can be employed to leverage the known clearance in workspace.

At any point p in workspace, we can define a spherical sampling region r with clearance awareness by considering the available space to place the robot. Let the center be p and radius be defined as the clearance at p minus the robot's minimum radius from the reference point p_r defining its translational DOFs. Define r such that any samples generated will place p_r within the sphere. Such a region r circumscribes the maximum region of workspace around p where a sample could be placed for p on the medial-axis.

In practice, few skeletons truly lie on the medial axis, and this sizing mechanism may preclude the generation of configurations at points where the skeleton's clearance is poor. To avoid this problem, a minimum radius should be considered to account for the fact that the clearance is not uniform around the skeleton components.

When working with a skeleton on the medial-axis, we can employ pure clearance-based sizing to filter out regions of workspace that cannot accept the robot. This enables the planner to avoid obstructions with small holes such as chain-link fences without wasting effort on an unfruitful exploration. In this case, the skeleton can be pruned of low-clearance points as a pre-processing step. We note that for skeletons not aligned with the medial-axis, biasing region selection by success rate still gives a strong preference for avoiding regions which are stuck at impassible regions of workspace.

Algorithm 1 Roadmap construction with Dynamic Region PRM

```
Require: Skeleton $S = (S_V, S_E)$, Roadmap $G = (G_V, G_E)$
 1: function BuildRoadmap( )
 2:     ***Initialize components at each skeleton vertex***
 3:     for all $v \in S_V$
 4:         $r \leftarrow$ GetRegionRadius($v$.point)
 5:         $Q \leftarrow$ SampleValidConfigurations($\beta_r(v)$)
 6:         $E \leftarrow \varnothing$
 7:         for all $q \in Q$
 8:             $N \leftarrow$ NearestNeighbors($q$, $Q$)
 9:             $E \leftarrow E \cup$ AttemptConnections($q$, $N$)
10:         for all Connected Componet $cc \in (Q, E)$
11:             InitializeRegions($cc$.vertices, $v$)
12:         $G_V \leftarrow G_V \cup Q$
13:         $G_E \leftarrow G_E \cup E$
14:     ***PRM Loop***
15:     while $\neg done$                    ▷ either node limit or $S$ covered
16:         $r \leftarrow$ SelectRegion()
17:         if $r \neq \varnothing$
18:             $Q \leftarrow$ ExpandLocalComponent($r$)
19:             while $\exists q \in Q \mid r$.Contains($q$)
20:                 AdvanceRegion($r$, $Q$)
21:             ConnectLocalComponents($r$.edge)
22:         else
23:             $e \leftarrow$ RandomUnconnectedSegment( )
24:             ConnectLocalComponents($e$)
```

C. Answering Queries

When presented with a query consisting of a start and a goal configuration $q_{start}, q_{goal} \in \mathcal{C}_{free}$, it is possible that either start or goal is not connectable to the current roadmap. This represents a case where either the skeleton missed the corresponding parts of workspace (resulting in no configurations nearby) or the nearby configurations lie in a region of $\mathcal{C}_{free}$ that is locally disconnected from q. The repair strategy is to expand rapidly outward from q in search of either the roadmap (thus completing the connection) or the skeleton (thus allowing the use of dynamic region guidance to complete the connection). An RRT is ideal for this purpose as it handles both cases elegantly: it will rapidly find a nearby skeleton point, either leading to a connection or arrival at a region near a skeleton vertex where dynamic region guidance can be employed. This is analogous to the Spark PRM strategy [26] where an RRT is locally employed to bridge narrow passages for a PRM planner.

Algorithm 2 Component expansion and connection

```
Require: Roadmap $G = (G_V, G_E)$
 1: ***Expand a local component***
 2: function ExpandLocalComponent(Region $r$)
 3:     $C_r \leftarrow$ GetLocalComponent($r$)
 4:     $Q \leftarrow$ SampleValidConfigurations($r$)
 5:     for all $q \in Q$
 6:         $N \leftarrow$ NearestNeighbors($q$, $C_r$)
 7:         $E \leftarrow$ AttemptConnections($q$, $N$)
 8:         if $E = \varnothing$
 9:             $Q \leftarrow Q \setminus \{q\}$
10:             continue                              ▷ couldn't connect
11:         $G_V \leftarrow G_V \cup q$
12:         $G_E \leftarrow G_E \cup E$
13:     $r$.UpdateSuccessRate($|Q|$, $K$)
14:     return $Q$
15: ***Connect local components in a segment***
16: function ConnectLocalComponents(SkeletonEdge $e$)
17:     ***Sample at a random point on the edge***
18:     $p \leftarrow$ RandomPoint($e$.path)
19:     $r \leftarrow$ GetRegionRadius($p$)
20:     $Q \leftarrow$ SampleValidConfigurations($\beta_r(p)$)
21:     ***Attempt to merge components***
22:     $C \leftarrow$ GetLocalComponents($e$)
23:     for all $q \in Q$
24:         $E \leftarrow$ AttemptConnections($q$, $C$)
25:         if $E$ has edges to more than one $c \in C$
26:             merge all $c \in C$ connected by $E$
```

Algorithm 3 Dynamic Region operations

```
 1: ***Initialize regions and local components***
 2: function InitializeRegions(Configurations $Q$, SkeletonVertex $v$)
 3:     for all $e \in v$.GetOutboundEdges( )
 4:         $C \leftarrow$ MakeLocalComponent($e$, $v$, $Q$)
 5:         CreateRegion($C$)
 6: ***Advance an expansion region one step***
 7: function AdvanceRegion(Region $r$, Configurations $Q$)
 8:     if $r$.AtEdgeEnd( )
 9:         ***Attempt to merge components***
10:         $C \leftarrow$ GetLocalComponents($r$.edge.target)
11:         for all $q \in Q$
12:             $E \leftarrow$ AttemptConnections($q$, $C$)
13:             if $E \neq \varnothing$
14:                 merge all $c \in C$ connected by $E$
15:         if not merged
16:             InitializeRegions($Q$, $r$.edge.target)
17:         DeleteRegion($r$)
18:     else
19:         ***Move to the next position***
20:         $r$.center $\leftarrow r$.GetNextSkeletonEdgePoint( )
21:         $r$.radius $\leftarrow$ GetRegionRadius($r$.center)
```

IV. Theoretical Properties

The algorithm can be expected to work well when the union of all sampling regions covering the skeleton points contains a path-connected volume in $\mathcal{C}_{free}$.

Formally, define the metric space $M_C = (\mathcal{C}_{space}, D)$ where D is some metric and $M_W = (W, T)$ where W is the workspace and T is translational distance. Let $\beta_r^C(q)$ be a ball in M_C of radius r centered at $q \in \mathcal{C}_{space}$, and let

$\beta_r^C(p)$ be a ball in W of radius r centered at $p \in W$. Let $\tau(q) : \mathcal{C}_{space} \rightarrow W$ represent the mapping between the translational subspace of $\mathcal{C}_{space}$ and the robot's reference point in W. Let $\tau(q) = p$ so that the image of $\{\tau(x) | x \in \beta_r^C(q)\} = \beta_r^W(\tau(q)) \subseteq W$. Since τ is a many-to-one function, the inverse image $Q = \{\tau^{-1}(x) | x \in \beta_r^W(p)\} \subseteq \mathcal{C}_{space}$ describes a hypercylinder of maximum height in $\mathcal{C}_{space}$ centered on $\tau^{-1}(p)$ such that $\beta_r^C(q) \subseteq Q$. This implies that a ball of radius r in M_W encompasses a superset of the corresponding ball of radius r in M_C. The union of all possible sampling regions along the skeleton U thus represent a union of hypercylinders in $X \subseteq \mathcal{C}_{space}$; in the case where U is path-connected, X will be also.

To ensure that U contains a path-connected volume in $\mathcal{C}_{free}$, it remains to show that the intersection $Y = X \cap \mathcal{C}_{free}$ is path-connected. A general argument for this is not possible due to the wide variety of choices for the skeleton, robot, and environment. For example the workspace and skeleton may be disjoint (in which case no planner can succeed in completely connecting the space), the skeleton may be badly positioned (resulting in disjoint components for Y), or the robot may be too large to traverse into certain regions of workspace (again resulting in disjoint components for Y). As such, this description serves as a characterization of when the method can produce a good coverage of W and $\mathcal{C}_{free}$ rather than a statement that it will always do so.

There are at least two cases where one can be assured that Y is path-connected. The first is when the robot is a free-body with maximum radius less than or equal to some value ρ and the skeleton has clearance greater than or equal to ρ everywhere. This holds for some common cases where floor-dwelling robots must perform tasks in large but reasonably uncluttered spaces. A second case is where there is a valid configuration at each skeleton point and a valid local plan between them: this represents the case where one knows that the robot has a valid maneuver for all localities, and shows a similar flavor to human intuition in collaborative planning [21].

In cases where Y is path-connected, the algorithm is probabilistically complete if the skeleton meets the definition of a deformation retract. This is true because a retract skeleton is visible to the entire $\mathcal{C}_{free}$, and a path between any two points can be formed by connecting each point to its nearest visible point on the skeleton.

V. Validation

To evaluate Dynamic Region sampling with PRM, we tested the method on two problems with multiple path homotopy classes and compared against PRM (baseline), PRM with Workspace Importance Sampling (WIS-PRM) [17], and Dynamic Region-biased RRT (DRB-RRT).

The environments include a `Garage` problem with a quadcopter robot, a `DhaA` protein with a ligand probe, and a cramped three-dimensional `GridMaze`. Each environment exhibits winding tunnels which increase the difficulty of connecting configurations. In each problem, the PRM planners build an initial roadmap with a fixed number of vertices before being presented with a series of queries. They then search for a solution for each query in sequence, starting from the current roadmap and expanding it if necessary. This exercises the multi-query intention of PRM and shows how well the constructed roadmaps generalize over several planning requests. The Dynamic Region-biased RRT method is included to contrast the performance of a guided single-query method.

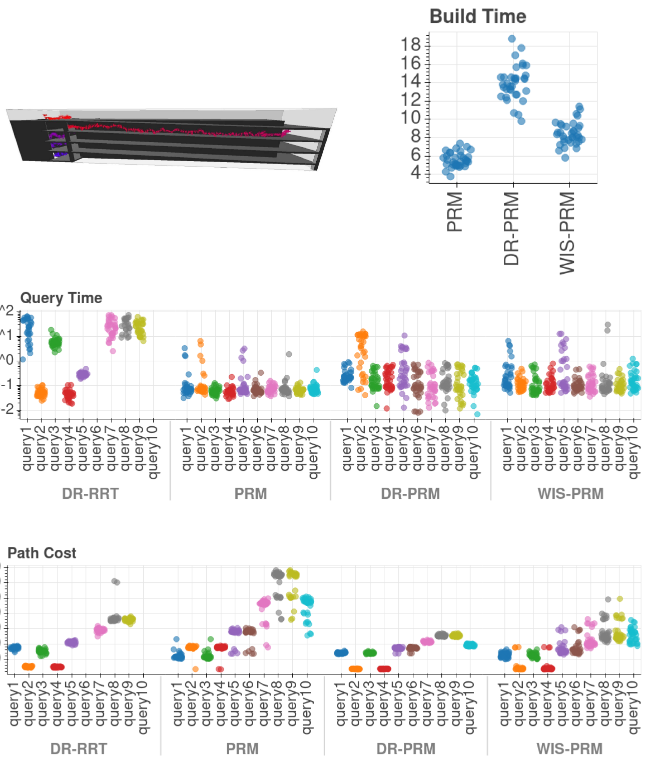


Fig. 3. `Garage` environment and experiment results. Time is reported in seconds. DRB-RRT failed to solve queries six and ten at all within an 80 second time limit, and occasionally failed queries one (four fails), eight (ten fails), and nine (three fails).

A. Experiment Setup

All methods were implemented in a C++ motion planning library developed in the Parasol Lab at Texas A&M University. All experiments were executed on a desktop computer running CentOS 7 with an Intel® Core™ i7-3770 CPU at 3.4 GHz, 16 GB of RAM, and the GNU g++ compiler version 4.8.5. Skeletonization for the dynamic region methods was performed with a Mean Curvature Skeleton [24] implemented in the CGAL library [27]. Workspace tetrahedralization for WIS-PRM was performed with a combination of the TetGen [28] and CGAL libraries. Time to build these models was considered preprocessing and not included in the result plots.

Each experiment ran until all queries solved. We performed 35 trials for each experiment and report the initial roadmap construction time, time to solve each query, and cost of the produced paths. Construction and Query time are reported in seconds, while path cost is in euclidean distance in $\mathcal{C}_{space}$. Each trial is plotted as a scatter dot to illustrate the spread of behavior.

PRM and WIS-PRM sample ten configurations per iteration in all environments. Dynamic Region PRM uses five in `Garage` and `Gridmaze` because it generates all samples for an iteration within the same locality and additionally attempts a second set of samples during the connection phase. In `DhaA` it uses ten samples to reflect the greater difficulty of sampling a valid configuration for the ligand probe. All PRM methods use eight nearest-neighbors.

B. Analysis

The `Garage` problem (Fig. 3) presents a series of ten queries scattered across the levels of the structure. The space is relatively open compared to the robot size, and the primary sources of difficulty are thin walls and large scale. We observe that Dynamic Region PRM consistently takes longer to construct an initial map than PRM or WIS-PRM, and takes longer to solve the second query. However, it consistently produces very low path costs with little variance. This occurs because the region guidance forces the planner to cover a regular volume around the skeleton edges, which provides a roadmap with paths that roughly map to paths through the skeleton (plus any distance needed to reach the skeleton if the query is far away). WIS-PRM produces better paths than PRM by taking greater care to sample in less accessible regions of workspace, thus providing coverage that is better but not as consistent as Dynamic Region PRM. Dynamic Region-biased RRT can sometimes match its PRM counterpart's path cost, but always takes longer to do so and frequently fails to find a path within a reasonable time limit (80 seconds here).

The `DhaA` problem (Fig. 4) presents a sequence of four queries representing ligand binding sites. The first three queries have start and goal positions close to the skeleton, while the fourth is farther away. We see that Dynamic Region PRM has the fastest build time, although the advantage is not highly significant over WIS-PRM. Its query time however is consistently low, whereas the other methods exhibit a significant spread of times. Path cost is better than PRM but not as low as WIS-PRM; this occurs because the Dynamic Region paths follow the skeleton closely, while the WIS-PRM paths hug the boundary relatively closely. The path for the fourth query is longer for Dynamic Region PRM because it lies farther from the skeleton. Here the nearest nodes are concentrated around the skeleton, so the path effectively 'snaps' to the skeleton's topology. This case illustrates a possible negative side-effect of skeleton guidance. Dynamic Region-biased RRT exhibits a similar issue with lower intensity due to constant-sized regions. However, its query time is subject to long-running outliers when the algorithm gets stuck trying to break through a low-clearance area.

The `Gridmaze` problem (Fig. 5) presents a sequence of four queries dispersed in the maze. The maze is fairly tight, making the entire workspace relatively close to the skeleton. In this setting Dynamic Region PRM excels with rapid build and query times compared with the other methods. Its path cost is also consistent and minimal, which is expected given the close matching between the workspace and its skeleton. This is an ideal case for Dynamic Region PRM, even over its

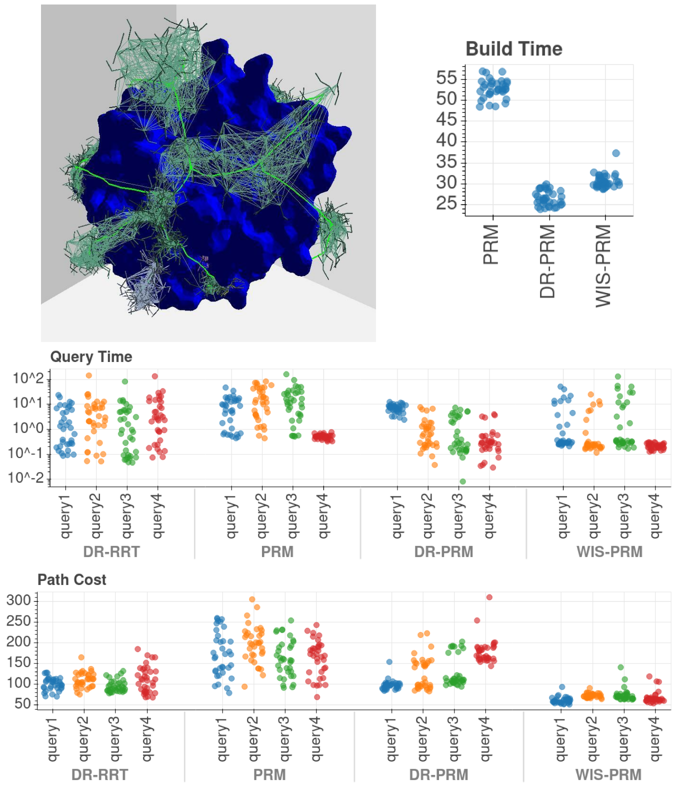


Fig. 4. `DhaA` environment and experiment results. Time is reported in seconds.

RRT counterpart which fails to discover the cheapest path for the first query.

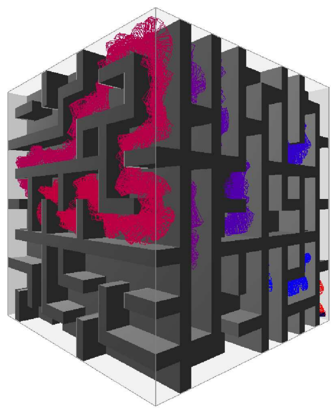
VI. Conclusion

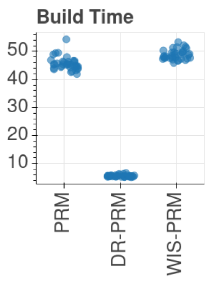
We present dynamic region sampling for sampling-based planners which create multiple connected components such as PRM, and show that it provides effective solutions in both robotics and ligand-binding applications. The method draws on a relationship between a workspace skeleton and paths in $\mathcal{C}_{free}$, and can be expected to perform well in problems where this relationship holds.

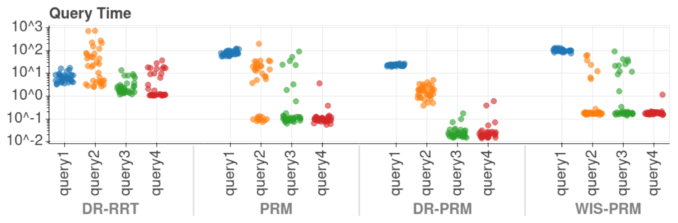
The most important future work is to demonstrate that dynamic region sampling can be applied to parallel PRM. This is especially important for protein folding applications where feasibility planning is very time consuming, and in applications such as binding-site evaluation where the goal is to discover as many valid paths as possible. These problems require large roadmaps in difficult spaces, yet the dynamic region PRM approach could be used to parallelize this process in a theoretically novel way. The skeleton edge segments imply a topology-induced partitioning of the problem, and the method presented here for bridging local connected components would be useful in combining solutions to partitioned sub-problems.

References

[1] B. Englot and F. S. Hover, "Sampling-based coverage path planning for inspection of complex structures," in *Proc. Int. Conf. Automated Planning and Scheduling*, 2012.







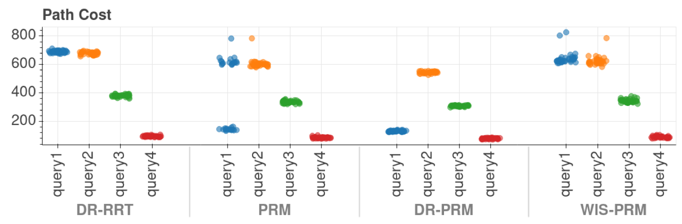


Fig. 5. `Gridmaze` environment and experiment results. Time is reported in seconds.

[2] D. Brutlag, M. Apaydin, C. Guestrin, D. Hsu, C. Varma, A. Singh, and J.-C. Latombe, "Using robotics to fold proteins and dock ligands." in *Bioinformatics*, 2002, p. 18:S74.S83.

[3] 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.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[4] A. akbar Agha-mohammadi, S. Agarwal, A. Mahadevan, S. Chakravorty, D. Tomkins, J. Denny, and N. M. Amato, "Robust online belief space planning in changing environments: Application to physical mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Hong Kong, May 2014.

[5] S. Thomas, X. Tang, L. Tapia, and N. M. Amato, "Simulating protein motions with rigidity analysis," *J. Comput. Biol.*, vol. 14, no. 6, pp. 839–855, 2007, special issue of Int. Conf. Comput. Molecular Biology (RECOMB) 2006.

[6] D. Hsu, J.-C. Latombe, and H. Kurniawati, "On the probabilistic foundations of probabilistic roadmap planning," *Int. J. Robot. Res.*, vol. 25, pp. 627–643, July 2006.

[7] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "OBPRM: an obstacle-based PRM for 3D workspaces," in *Proc. Int. Wksp. Alg. Found. Robot. (WAFR)*. Natick, MA, USA: A. K. Peters, Ltd., 1998, pp. 155–168.

[8] J. Denny and N. M. Amato, "Toggle PRM: A coordinated mapping of C-free and C-obstacle in arbitrary dimension," in *Alg. Found. Robot. X*. Springer, 2013, pp. 297–312, (WAFR '12).

[9] J. Denny, R. Sandström, A. Bregger, and N. M. Amato, "Dynamic region-biased exploring random trees," in *Alg. Found. Robot. XII*. Springer, 2020, (WAFR '16).

[10] R. Sandström, "Approximating configuration space topology with workspace models," Ph.D. dissertation, Texas A&M University, 2020.

[11] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, Oct. 1979.

[12] J. H. Reif, "Complexity of the mover's problem and generalizations," in *Proc. IEEE Symp. Found. Comp. Sci. (FOCS)*, San Juan, Puerto Rico, Oct. 1979, pp. 421–427.

[13] V. Boor, M. H. Overmars, and A. F. van der Stappen, "The Gaussian sampling strategy for probabilistic roadmap planners," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 2, May 1999, pp. 1018–1023.

[14] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 2, 1999, pp. 1024–1031.

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

[16] M. Morales, L. Tapia, R. Pearce, S. Rodriguez, and N. M. Amato, "A machine learning approach for feature-sensitive motion planning," in *Alg. Found. Robot. VI*. Springer, 2005, pp. 361–376, (WAFR '04).

[17] H. Kurniawati and D. Hsu, "Workspace importance sampling for probabilistic roadmap planning," in *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, vol. 2, Sept. 2004, pp. 1618–1623.

[18] J. P. van den Berg and M. H. Overmars, "Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners," *Int. J. Robot. Res.*, vol. 24, no. 12, pp. 1055–1071, 2005.

[19] H. Kurniawati and D. Hsu, "Workspace-based connectivity oracle - an adaptive sampling strategy for prm planning," in *Alg. Found. Robot. VII*. Springer, 2008, pp. 35–51, (WAFR '06).

[20] E. Plaku, L. Kavraki, and M. Vardi, "Motion planning with dynamics by a synergistic combination of layers of planning," *IEEE Trans. Robot.*, vol. 26, no. 3, pp. 469–482, June 2010.

[21] J. Denny, R. Sandström, N. Julian, and N. M. Amato, "A region-based strategy for collaborative roadmap construction," in *Alg. Found. Robot. XI*. Springer, 2015, pp. 125–141, (WAFR '14).

[22] S. Bhattacharya, M. Likhachev, and V. Kumar, "Topological constraints in search-based robot path planning," *Autonomous Robots*, vol. 33, no. 3, 2012.

[23] H. Blum, "A transformation for extracting new descriptors of shape," in *Models for the Perception of Speech and Visual Form*, W. Wathen-Dunn, Ed. MIT Press, 1967, pp. 362–380.

[24] A. Tagliasacchi, I. Alhashim, M. Olson, and H. Zhang, "Mean curvature skeletons," in *Proc. Symp. Geom. Proc.*, vol. 31, no. 5, 2012, pp. 1735–1744.

[25] L. Shapira, A. Shamir, and D. Cohen-Or, "Consistent mesh partitioning and skeletonisation using the shape diameter function," *The Visual Computer*, vol. 24, no. 4, pp. 249–259, 2008.

[26] K. Shi, J. Denny, and N. M. Amato, "Spark PRM: Using RRTs within PRMs to efficiently explore narrow passages," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Hong Kong, P. R. China, June 2014.

[27] A. Fabri, G.-J. Giezeman, L. Kettner, S. Schirra, and S. Schönherr, "On the design of CGAL a computational geometry algorithms library," *Softw. – Pract. Exp.*, vol. 30, no. 11, pp. 1167–1202, 2000.

[28] H. Si, "Tetgen, a delaunay-based quality tetrahedral mesh generator," *ACM Trans. Math. Softw.*, vol. 41, no. 2, pp. 11:1–11:36, Feb. 2015. [Online]. Available: http://doi.acm.org/10.1145/2629697