

Using Local Experiences for Global Motion Planning

Constantinos Chamzas¹, Anshumali Shrivastava¹, Lydia E. Kavraki¹

Abstract—Sampling-based planners are effective in many real-world applications such as robotics manipulation, navigation, and even protein modeling. However, it is often challenging to generate a collision-free path in environments where key areas are hard to sample. In the absence of any prior information, sampling-based planners are forced to explore uniformly or heuristically, which can lead to degraded performance. One way to improve performance is to use prior knowledge of environments to adapt the sampling strategy to the problem at hand. In this work, we decompose the workspace into local primitives, memorizing local experiences by these primitives in the form of local samplers, and store them in a database. We synthesize an efficient global sampler by retrieving local experiences relevant to the given situation. Our method transfers knowledge effectively between diverse environments that share local primitives and speeds up the performance dramatically. Our results show, in terms of solution time, an improvement of multiple orders of magnitude in two traditionally challenging high-dimensional problems compared to state-of-the-art approaches.

I. INTRODUCTION

Motion planning is an integral part of many areas of robotics. Robots operating autonomously need to generate many different motion plans in complex environments. This is true especially in the context of task and motion planning [1]. Even a single task, such as stacking blocks, might require querying a motion planner thousands of times. Humans can execute motions instantaneously that robots currently struggle with. To achieve human-level behavior, fast online motion planning is essential.

The widespread success of sampling-based planners lies in their ability to approximate the connectivity of high-dimensional spaces with a small number of samples [2]. However, in many cases regions necessary for connectivity are unlikely to be sampled by an uninformed sampler. This is known as the *narrow passages* problem [3], [4], [5] and greatly limits the performance of sampling-based planners in many scenarios.

Among the possible approaches to solving problems that involve narrow passages is the emerging field of experience-based planning [6], [7], [8]. It is common for robots during their operation to come across similar workspaces resulting in similar motion plans. Such a case can be seen in Fig. 1 where a robot needs to grasp the red can and put it on the top shelf. In this case, prior knowledge about similar scenarios could expedite the motion planning process. By biasing sampling towards interesting regions [7] or by retrieving and reusing old solutions [6],



Fig. 1: The task of grasping the red can without removing the green cans is very challenging for traditional methods. The proposed approach efficiently solves it by using prior experiences to guide the search.

experience-based methods try to transfer knowledge obtained from similar problems to others. Unfortunately, small changes in the workspace can drastically affect the possible solutions, in several cases, thus making generalization difficult [9], [10].

This paper presents a sampling strategy for sampling-based planners that aids in discovering the connectivity of the configuration space even for pathological cases. This sampling strategy utilizes a decomposition of the workspace into local primitives. The main insight of our method is that learning to generate important samples in the configuration spaces defined by the robot and the primitives helps approximate the configuration space of the global workspace. This approach can generalize to new environments that contain the workspace primitives used earlier or primitives very similar to them. In this work, we focus on problems with simple geometric features, yet manage to solve a class of problems that were practically unsolvable by modern sampling-based motion planners. Also, we raise the question of whether the notion of decomposition applies in unstructured environments with complex geometries, tackling problems that were previously beyond the reach of sampling-based motion planners.

The contributions of this paper are threefold. First, we propose the decomposition of the workspace into local primitives, and we solve motion planning problems in workspaces that contain only the local primitives. The result of this step is the estimation of local samplers that produce samples in the difficult regions of the configuration spaces of the local primitives. The parameters of these local samplers are stored in a database. Second, we show how to synthesize a global sampling strategy based on these local samplers. Third, we show the effectiveness of our approach in two challenging environments where it achieves significant improvement over existing methods.

*This work has been supported in part by NSF 1718478 and Rice University Funds.

¹ Department of Computer Science, Rice University, Houston TX, USA {chamzas, anshumali, kavraki}@rice.edu

II. BACKGROUND

Sampling-based planners have been widely adopted in robotics due to their ability to scale to high-dimensional problems. The two main categories are graph-based approaches (e.g., PRM [11]) for multi-query problems and tree-based (e.g., RRT [12], EST [13]) for single-query problems. Although in general motion planning is PSPACE-complete [14], [15], sampling-algorithms perform remarkably well. However, in challenging environments (e.g., instances with narrow passages), sampling plays an increasingly important role in the planning performance. It is theoretically understood, however, that using non-uniform samplers to generate more samples in low-expansiveness areas [13] can alleviate this problem.

Many approaches use rejection sampling, where the sample is accepted only if it passes a specific geometric test such as Bridge-Sampling [3] or Gaussian-Sampling [16]. Other approaches try to guess difficult areas, such as Medial-Axis sampling [17] or Obstacle-Based Sampling [5]. Although these approaches ultimately generate samples near or inside narrow passages, they still consider the entire configuration space which is computationally expensive. Nevertheless, the resulting graph is typically much smaller than the one produced by uniform sampling.

Recent approaches similar to our method try to leverage acquired information about the problem, to bias the sampling. Reinforcement learning was used by [18] to infer important areas of the workspace that were transformed to configuration samples. The authors of [7] utilize a generative model, called Conditional Variational Auto-Encoder (CVAE), that learns to produce samples that lie in “interesting” areas given a workspace description. Sampling-biasing methods have the advantage that they can be used with many sampling-based planners without any modification. However, both of these methods rely on a model that uses workspace features to infer important samples in the configuration space. In [18], a discrete workspace cell was mapped to a configuration through the inverse kinematics of the end-effector. In [7], a neural network was used to infer these samples. Our experiments showed that in complex configuration spaces these models do not consistently produce samples in the important areas of the configuration space.

Online-adaptive sampling methods use collision checking information to infer which areas of the configuration space are important at runtime. Utility-guided sampling [19] chooses samples with the maximum information gain based on the entropy of the roadmap. The authors of [20] formulate the sampling problem as a classification between free and in-collision samples. Toggle-PRM [21] creates two roadmaps, one in the configuration space and one in the obstacle space, and tries to infer samples in the narrow passages. These methods adapt the sampling online, based on the state the motion planning algorithm, but do not transfer this knowledge between different planning queries. On the contrary, the proposed method biases the sampler based on previous planning queries. Thus, online-adaptive sampling can be used in parallel with the proposed sampling biasing.

Orthogonal to these methods are database approaches. Instead of modifying sampling, these methods leverage previous experiences by storing discrete paths or graphs in a database. This information is later retrieved and repaired/transformed to satisfy the new kinematic constraints. These methods can be thought of as hard-coded experiences compared to biasing the sampling.

The authors of [22] used a library of paths that was queried based on the proximity of start-goal configurations of the entry and the query. If a valid path could not be retrieved based on these heuristics, BIRRT [12] was used to repair the invalid parts. This idea was expanded in [6] where a graph was used to store the paths removing redundancy. Although improving the execution time by orders of magnitude, the mentioned approaches do not adapt well to changes and yield improved results mainly in invariant environments. This happens because they do not explicitly include the workspace in their experience representation.

Database approaches that explicitly use workspace information include [23] which creates a small database of obstacle road-maps smartly decomposing the configuration space in obstacle-maps. The trajectory prediction proposed by [24] saves the generated paths in task-space and during execution transforms them back to the configuration space by optimizing the cost of the trajectory while using IK. Although these methods can deal with different environments [23] works only for free-flying robots and [24] works with trajectory optimization planners that lack the probabilistic guarantees of sampling-based planners.

In this work, we combine the best of both worlds by integrating a biased-sampler with a database. This is achieved by decomposing the workspace in local primitives and storing in a database the parameters of an efficient local sampler for each local primitive. The biased-sampler uses prior knowledge in a “soft way” avoiding the hard-commitment to complete paths, induced by databases methods, which may need costly repairing when there are significant changes in the workspace. On the other hand, the database-scheme enables the instant mapping of local samplers to local primitives avoiding the need for a complex parametric model. Additionally, a database has the inherent capability of incrementally improving its experiences by simply adding new entries. The aforementioned sampling-biasing approaches would need to be retrained.

III. METHOD OVERVIEW

In this work, we modify the sampling step that produces configuration samples, which is at the core of all sampling-based planners as shown in Algorithm 1. Similar to [7] we generate λN samples from the global sampler (line 3) and $(1 - \lambda)N$ from the uniform sampler (line 5). λ is a hyperparameter that is determined by the application. The data structure G and the `update()` function (line 7) differentiates the sampling-based planners. For example in a PRM-like setting, G is a roadmap, and new samples are added to expand the roadmap which captures the connectivity of the C-Space in a variety of ways [11], [21]. In an RRT-like setting, G is a

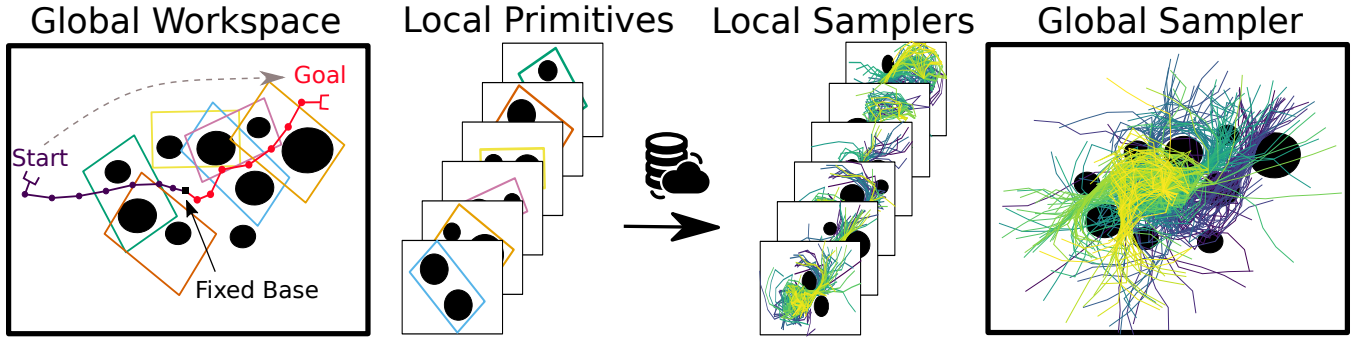


Fig. 2: An illustration of our proposed method. In this example, the workspace contains circular obstacles and a planar arm with a fixed base. In our approach, the workspace is decomposed to local primitives. During the offline phase, the local samplers are computed and saved in the database. During the online phase, each local primitive is mapped to its corresponding local sampler through the database. Finally, the local samplers are synthesized to a global sampler.

tree, and the sample is used to expand the tree in different ways [12], [21]. However, as stated above, regardless of the planner details, the performance can be improved by using informed sampling.

Algorithm 1: Modified Sampling Procedure

Input : Number of iterations N
Output : Graph structure G

```

1 while  $i \leq N$  or solutionFound() do
2   if  $\text{rand}(0,1) \leq \lambda$  then
3      $x \sim \text{GL-Sampler}()$ 
4   else
5      $x \sim \text{Uniform}()$ 
6   end
7    $\text{update}(G, x)$ 
8 end
9 return  $G$ 

```

The proposed sampling strategy is based on local primitives of the workspace. The key insight of our approach is that different regions of the workspace often create relatively uncorrelated narrow passages in the configuration space. For example, the top and bottom shelf of a bookcase will create two different challenging regions that the robot arm will need to traverse when moving from one shelf to the other. This means that we can bind an effective local sampling distribution to each shelf (local primitive) dealing with the two problems independently. By combining the local samplers, we can synthesize a global sampler that informatively guides the planners. Our experiments showed that even in highly correlated cases, our approach is still effective.

We will introduce some notation, before describing the main algorithm. Since our method relies on a workspace description, we will denote the workspace as $W = \{wo_1, \dots, wo_M\}$ where each wo_i is a workspace obstacle. Also we denote the set of local primitives as $LW = \{lw_1, \dots, lw_N\}$, such that $\bigcup_{i=1}^N lw_i \subseteq W$. Note that the local primitives do not have to be mutually exclusive. For example, pairs of workspace obstacles $lw_i = \{wo_j, wo_k\}$ could be a valid set of local primitives. The global target

distribution is denoted as $p(x|W)$. Local samplers that are used to approximate it are denoted as $p_i(x|lw_i)$.

The main steps of the Global-Local Sampler (GL-Sampler) are outlined in Algorithm 2. The first step (line 1) is to decompose the workspace W by identifying its local primitives LW . In general, this could be a highly sophisticated function; however, in the context of this work, the local primitives are always pairs of workspace obstacles $lw_i = \{wo_j, wo_k\}$. For each of the local primitives, we try to retrieve their corresponding samplers (line 4) based on a similarity function k (line 3). If no similar local primitive exist in the database, then the parameters of the local sampler are calculated (line 6) and stored in the database (line 7). Note that the database can also be computed offline if the possible local primitives are known before-hand which is often the case. The local samplers are combined to synthesize the global sampler (line 10). The GL-Sampler is created only at the beginning of the motion planning query.

Algorithm 2: Create GL-Sampler

Input : workspace W , database D , threshold d
Output : GL-Sampler()

```

1 Decompose  $W$  to  $LW = \{lw_1 \dots lw_N\}$ 
2 foreach  $lw_i \in LW$  do
3   if  $\exists lw_j \in D$  s.t.  $k(lw_j, lw_i) < d$  then
4     Retrieve  $p_i(x|lw_i)$  from  $D(lw_i)$ 
5   else
6     Create  $p_i(x|lw_i)$ 
7     Insert  $lw_i : p_i(x|lw_i)$  in  $D$ 
8   end
9 end
10 Synthesize  $\hat{p}(x|W)$  from  $p_i(x|lw_i)$ 
11 return  $\hat{p}(x|W)$ 

```

Fig. 2 shows an illustrating example of the algorithm where it is applied on a fixed-base, 8-link planar manipulator (non-intersecting) in an environment of circular obstacles. The local primitives in this case are pairs of circles and are described as $lw_i = \{x_a, y_a, r_a, x_b, y_b, r_b\}$, where x, y, r denote the position and radius of each circle. In the following

sections, the creation of the database, retrieval of local sampler and composition of the global sampler are described.

A. Creating the Database of Local Samplers

Each local sampler $p_i(x|lw_i)$ must produce samples that quickly capture the local primitive's configuration space. For example, each local sampler should produce samples in narrow passages of the corresponding C-Space. First, we generate such samples and later fit the local sampler to them. Such samples are created by solving motion planning queries that likely traverse difficult regions of the configuration space of the local primitives. For every local primitive we pre-specify a set of such motion planning queries. For the local primitives (pairs of circles) in Fig. 2, a path that starts with the robot between the circles and ends with the robot entirely out of them likely traverses a narrow passage. A standard sampling planner e.g., RRT, BIRRT [12] is used to solve these queries quickly. Additionally, it is imperative that multiple paths for the same query are generated to be robust to obstacles that are part of the global workspace but not the local primitive. This is clear in Fig. 2 where several samples which were valid for the local primitives are invalid for the global problem. To deal with this we run the chosen planner multiple times, creating different paths due to its randomness. Also by using shortcutting techniques [25], most of the redundant samples can be removed to increase the ratio of the useful samples.

Due to the complexity of the needed distribution and its natural multi-modality, we choose a Gaussian Mixture Model (GMM) as the local sampler similar to [10]. However, contrary to [10] we do not use the traditional expectation-maximization algorithm to calculate the parameters of the GMM. There is no good way to choose the number of mixtures, and more importantly, the distance between configurations does not necessarily relate to C-Space connectivity which is what sampling-based algorithms need to capture. Instead, for the local sampler i we choose to place one mixture to each produced configuration q_{i1}, \dots, q_{iM} and use a fixed covariance $\Sigma_i = \sigma I$ where σ is a hyperparameter and I is the identity matrix. This might create an unnecessarily large amount of mixtures, but we present a way to reduce them while respecting the connectivity in section IV-B. The local sampler will be:

$$p_i(x|lw_i) = \text{GMM}(\mu, \Sigma) = \frac{1}{M_i} \sum_{j=1}^{M_i} \mathcal{N}(q_{ij}, \Sigma_{ij}) \quad (1)$$

M_i is the number of mixtures. We choose a uniform vector for the weights making all the mixtures equiprobable. In the database, we save the parameters of this distribution and its corresponding local primitive.

B. Retrieving Local Samplers

To retrieve the local samplers from the database, we need a similarity function $k(lw_i, lw_k)$ to compare the local primitives between them. General workspace descriptors and possible similarity functions have been described by [24]. In our case

where the primitives are simple geometric descriptions the negative squared Euclidean distance is used:

$$k(lw_i, lw_j) = -(lw_i - lw_j)^T(lp_i - lp_j)$$

We retrieve all the parameters of local samplers that are above a certain threshold d of this similarity. Since the local samplers retrieved will not correspond to the exact local primitives a *similarity error* is introduced.

C. Synthesizing the Global Sampler

Now we will describe how the local samplers approximate the global sampler. Given an arbitrary partition of the configuration space $\bigcup_{i=1}^N X_i = X$, $X_i \cap X_j = \emptyset$, $\forall i, j$, $i \neq j$, the global sampler can be expressed as a sum of other distributions using the law of total probability:

$$p(x|W) = \sum_{i=1}^N p(x|W, X_i)p(X_i) = \sum_{i=1}^N p_i^*(x|W)a_i \quad (2)$$

In the last equation we rewrote $p(X_i)$ as a_i and $p(x|W, X_i)$ as $p_i^*(x|W)$. Note that the support of $p_i^*(x|W)$ is X_i . We approximate it in the following way:

$$p_i^*(x|W) \approx p_i(x|W) \approx p_i(x|lw_1, \dots, lw_N) \approx p_i(x|lw_i) \quad (3)$$

Three approximations are used in the derivation above. The first is one is that $p_i(x|W)$ has its support on X instead of X_i . This induces only a small error because $p_i(x|lw_i)$ is a distribution that has values close to zero outside X_i . The second one is that most of the information in W is incorporated in the set of local primitives $\{lw_1, \dots, lw_N\}$ which is true if the local primitives are responsible for most of the difficult regions of the configuration space. The final approximation is that the local primitives independently affect only one local distribution. $p_i(x|lw_1, lw_2, \dots, lw_N) \approx p_i(x|lw_i)$. This is not true in general especially in cases where the local primitives are close together. This is the reason why multiple paths are created for each local primitive in section III-A. We refer to this as the *decomposition error*. In the experiments section, we show empirically that even when this error is large our sampling method is much more effective than uniform random sampling. Finally combining Eq. 1, Eq. 2 and Eq. 3 the global sampler is:

$$p(x|W) \approx \sum_{i=1}^N \frac{a_i}{M_i} \sum_{j=1}^{M_i} \mathcal{N}(q_{ij}, \Sigma_{ij}) = \sum_{i=1}^N \frac{1}{N} \sum_{j=1}^{M_i} \mathcal{N}(q_{ij}, \Sigma_{ij})$$

We set $a_i = \frac{M_i}{N}$ which means that the weight of each local sampler is proportional to the number of its mixtures.

IV. REDUCING THE DATABASE

Since querying the database happens online, it is crucial to keep its size at a minimum for fast retrieval. We propose two such reductions. One is removing mixtures from the database by merging cliques, and the other is transforming the local experiences to account for multiple local primitives.

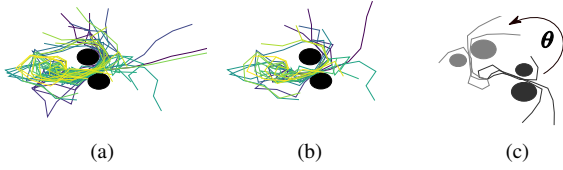


Fig. 3: (a) Un-merged Cliques, 53 mixtures needed (b) Merged Cliques, 23 mixtures needed, (c) Transforming the local sampler of the black primitive to apply to they grey primitive.

A. Merging Cliques

This step can be executed after the generation of the useful configurations in section III-A. Using the extracted configurations q_{i1}, \dots, q_{iM} for a specific local primitive lw_i we create a graph by connecting two configurations with an edge if there is a collision-free straight line between them. From this graph, we identify the fully connected subgraphs, also known as cliques. The cliques essentially represent groups of configurations that can be accurately approximated with one mixture model. We are not interested in finding the optimal number of cliques which is an NP-HARD problem, and for that reason, we are using a greedy algorithm that finds them sequentially. After finding the cliques, we calculate the mean and the covariance (if enough samples exist) of each one and use them as a parameter to the GMM as in Eq. 1. Our experiments showed that the time performance was similar when using the merged or the un-merged cliques. An example before merging the cliques can be seen in Fig. 3a where the GMM model would have 53 mixtures, after merging Fig. 3b shows the GMM would have 23 mixtures which save a lot of space in the database.

B. Transforming the Local Experiences

To reduce the size of the database, we employ a transformation scheme similar to [23] that takes advantage of the inherent symmetries of the robot. This allows a local sampler $p_i(x|lw_i)$ to be transformed to $p'_i(x|lw'_i)$ where lw'_i is a transformed local primitive. Essentially the inherent symmetries of a robot are changes in the configuration of the robot that counter transformations of the local primitives like rotation or translation. Although this is not true in the general case, the majority of robots such as mobile manipulators, drones, and robotic arms have such symmetries. In the kinematic chain scenario Fig. 2, the inherent symmetry is its rotational invariance around its fixed base. This is illustrated in Fig. 3c where the grey local primitive is the rotated version of the black local primitive around the fixed base $lw_{grey} = lw_{black} * Rot(\theta)$. Applying this simple transformation to the first joint in the means of the GMM $\mu_{grey}^{j1} = \mu_{black}^{j1} + \theta$ the local distribution applies to the new local primitive.

This simple transformation significantly reduces storage requirements. In the studied example the dimensionality of $lw_i = \{x_a, y_a, r_a, x_b, y_b, r_b\}$ is 6D and the database must store representative local primitives/local sampler from this 6D space. However, by using the mentioned transformation,

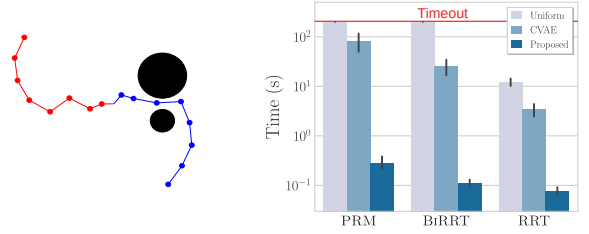


Fig. 4: Scenario with low decomposition and similarity error.

this space effectively becomes 5D, since we have rotational invariance of the local primitives around the fixed base.

V. EXPERIMENTS

In the following experiments, we compared different methods for the sampling part of three of the most representative sampling based planners, RRT, BiRRT [12], and PRM [11]. We benchmarked against uniform sampling, and the Conditional Variational Auto-Encoder (CVAE) proposed by [7]. The CVAE method is a neural network that is trained to produce samples that lie on the optimal path given a workspace description. To make the benchmarking fair we trained the CVAE using the same dataset that was used for the estimation of the GMMs. We used the OMPL [26] benchmarking tools [27] in their default settings, and run on an Intel i7 Linux machine with 4 GHz cores and 16GB of RAM. Each query was repeated 20 times, and the timeout was set at 200 seconds. In the figures where RRT is not shown it did timeout in all queries. Also, the uniform to biased ratio was chosen to be $\lambda = 0.5$, and the variance parameter was chosen to be $\sigma = 0.1$.

A. 8-link Kinematic Chain

Similar to [18] we used a fixed-based planer arm in an environment where obstacles are of varying sizes to demonstrate the strength of our approach. The kinematic chain had 8 links of variable lengths from 1 to 2 units and the circles variable radiuses from 1 to 2 units as well. The gap between most of the circles was less than 1 unit making it a very difficult problem. As local primitives, we consider only pairs of circles that are close together. Examples of such local primitives can be seen inside the colored rectangles of Fig. 2. We pre-computed the database such that any local primitive could be queried with a similarity error less than 3. Utilizing the transformation described in section IV-B only 800 pairs of obstacles were needed resulting in a small database and a very fast retrieval time (a few milliseconds). The total time for computing the database was around 10 minutes. We tested our method in three scenarios of scaling difficulty. The start configuration is noted with blue, and the goal configuration is noted with red. Note that we use log-scale for the time axis in all figures except Fig. 6.

1) *Scenario1*: The first scenario Fig. 4, contains only one local primitive thus having a *decomposition error* of zero and a low *similarity error* making it ideal for our method which found all the solutions in a fraction of a second. However, the trained CVAE did not succeed in approximating an efficient

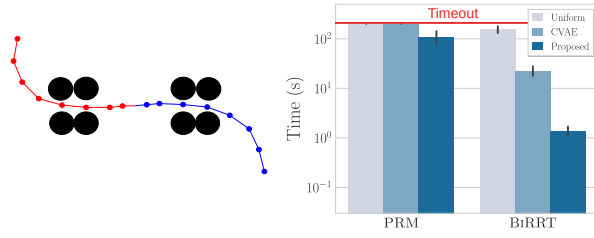


Fig. 5: Scenario with large decomposition error.

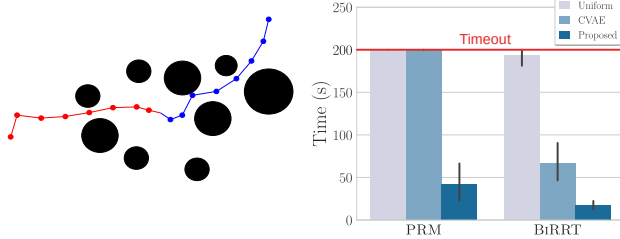


Fig. 6: Scenario with large decomposition and similarity error.

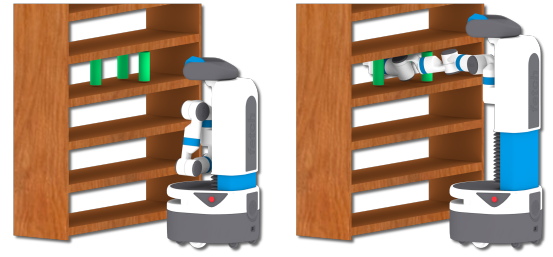
local sampler requiring an order of magnitude more time for the same problems. Finally, it can be seen that both PRM and BIRRT with uniform sampling did not solve any problems within 200s.

2) *Scenario2*: The second scenario Fig. 5, has a large *decomposition error* due to the proximity of the local primitives. Each local sampler is trained only on a pair of circles, and thus the majority of the samples produced are invalid when other local primitives are nearby. However, it can be seen that our method still significantly outperformed the other ones. Most notably BIRRT, with our method, found solutions in 1s while the CVAE needed around 20s and the Uniform around 200s. Note also that Uniform-PRM and CVAE-PRM timed-out in all cases. This scenario shows that the proposed method is very robust to approximation errors and can potentially work on very complicated environments that are decomposed into circles.

3) *Scenario3*: This scenario Fig. 6, is similar to the one used in [18] but much more difficult due to the closeness of the obstacles and the relatively large size of the robot. Both the *decomposition error* as well as the *similarity error* are large. In Fig. 2 the different local primitives and their local samplers which were used to create the global sampler are visible. Note that one circle is not part of any local primitive because it is not close to any other circle. Also in this scenario, our method outperformed the others with PRM succeeding only when using our method and BIRRT needing more than 10 minutes to find a solution with uniform sampling.

B. 8-DOF Robot

We also experimented on a simulated Fetch Robot [28] performing an object manipulation task. The robot has a 7-DOF arm and a moving torso resulting in an 8-DOF C-Space. The Fetch Robot tries to place its arm between the cylinders with start and goal shown in Fig. 7a, which is a difficult problem as it requires reaching into a deep shelf.



(a) Start Configuration (Left), Goal Configuration (Right).

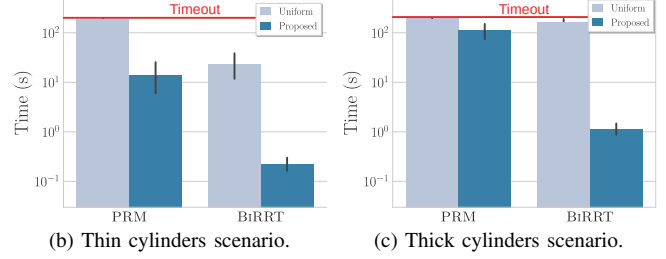


Fig. 7: Challenging motion planning problem and results for the Fetch Robot.

To demonstrate the practicality of this approach, we constructed a small database using only the local primitives that were present in the test scene. These local primitives were 3 pairs, each with one of the cylinders and the bookcase. Note that since each local primitive contains only 1 cylinder, it is significantly easier to solve than the full problem. We tested the proposed method on 2 scenarios. The first scenario has the same local primitives that existed in the database thus a *similarity error* of zero but a high *decomposition error*. The second scenario has thicker cylinders with a double radius which introduces a *similarity error*.

We only benchmarked against uniform sampling since there was not a rich dataset to train the CVAE. In the results, the increased difficulty of the second scenario is clearly visible. Both planners had zero or low success rate with uniform sampling while our approach succeeded in all of them.

VI. CONCLUSIONS

In this work, we proposed a new sampling-biasing framework that is based on a decomposition of the workspace. We only considered simple geometric primitives, yet we solved problems that were either not possible to solve with the methods we considered or not efficiently solved. Although we consider our results preliminary, we believe that this work paves a new way to apply experience in motion planning problems. Future work could include the use of more complicated primitives that are general and can effectively decompose any workspace. Finally, as the database grows in size efficient real-time retrieving algorithms should be used.

ACKNOWLEDGMENTS

The authors thank B. Willey, J. Hernandez, J. Abella, and M. Moll for their valuable and interesting conversations. The authors especially thank Z. Kingston for the visualization and benchmarking tools that made the Fetch experiment possible.

REFERENCES

- [1] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "An incremental constraint-based framework for task and motion planning," *The International Journal of Robotics Research (IJRR)*, vol. 37, no. 10, pp. 1134–1151, 2018.
- [2] H. M. Choset, S. Hutchinson, K. M. Lynch, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementation*. MIT press, 2005.
- [3] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3, pp. 4420–4426, 2003.
- [4] D. Hsu, L. E. Kavraki, J.-C. Latombe, R. Motwani, S. Sorkin, *et al.*, "On finding narrow passages with probabilistic roadmap planners," in *Robotics: The Algorithmic Perspective: Workshop on the Algorithmic Foundations of Robotics*, pp. 141–154, 1998.
- [5] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "OBPRM: An Obstacle-Based PRM for 3D Workspaces," in *Third Workshop on the Algorithmic Foundations of Robotics on Robotics: The Algorithmic Perspective*, pp. 155–168, 1998.
- [6] D. Coleman, I. A. Sucan, M. Moll, K. Okada, and N. Correll, "Experience-based planning with sparse roadmap spanners," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 900–905, June 2015.
- [7] B. Ichter, J. Harrison, and M. Pavone, "Learning Sampling Distributions for Robot Motion Planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7087–7094, May 2018.
- [8] M. Phillips, B. J. Cohen, S. Chitta, and M. Likhachev, "E-graphs: Bootstrapping planning with experience graphs," in *Robotics Science and Systems (RSS)*, July 2012.
- [9] B. Kim, L. P. Kaelbling, and T. Lozano-Perez, "Learning to guide task and motion planning using score-space representation," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2810–2817, 2017.
- [10] P. Lehner and A. Albu-Schaffer, "The Repetition Roadmap for Repetitive Constrained Motion Planning," *IEEE Robotics and Automation Letters (RAL)*, vol. 3, no. 3, pp. 3884 – 3891, 2018.
- [11] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation (TRA)*, vol. 12, no. 4, pp. 566 – 580, 1996.
- [12] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," *IEEE International Conference on Robotics and Automation (ICRA): Millennium Conference*, vol. 2, pp. 995–1001, 2000.
- [13] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *International Conference on Robotics and Automation (ICRA)*, vol. 3, pp. 2719–2726, 1997.
- [14] J. Canny, *The Complexity of Robot Motion Planning*. MIT press, 1988.
- [15] J. Canny, "Some algebraic and geometric computations in PSPACE," in *Annual ACM symposium on Theory of computing (STOC)*, pp. 460–467, 1988.
- [16] V. Boor, M. H. Overmars, and a. V. D. Stappen, "The Gaussian sampling strategy for probabilistic roadmap planners," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, pp. 1018–1023, May 1999.
- [17] O. Brock, "Adapting the sampling distribution in PRM planners based on an approximated medial axis," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 5, pp. 4405–4410, April 2004.
- [18] M. Zucker, J. Kuffner, and J. A. Bagnell, "Adaptive workspace biasing for sampling-based planners," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3757–3762, 2008.
- [19] B. Burns and O. Brock, "Toward optimal configuration space sampling," in *Robotics Science and Systems (RSS)*, pp. 105–112, 2005.
- [20] O. Arslan and P. Tsiotras, "Machine learning guided exploration for sampling-based motion planning algorithms," *International Conference on Intelligent Robots and Systems (IROS)*, pp. 2646–2652, December 2015.
- [21] J. Denny and N. M. Amato, "Toggle PRM: A coordinated mapping of C-free and C-obstacle in arbitrary dimension," *Springer Tracts in Advanced Robotics*, vol. 86, pp. 297–312, 2013.
- [22] D. Berenson, P. Abbeel, and K. Goldberg, "A robot path planning framework that learns from experience," *International Conference on Robotics and Automation (ICRA)*, pp. 3671–3678, 2012.
- [23] J.-M. Lien and Y. Lu, "Planning motion in environments with similar obstacles," *Robotics Science and Systems (RSS)*, 2009.
- [24] N. Jetchev and M. Toussaint, "Fast motion planning from experience: trajectory prediction for speeding up movement generation," *Autonomous Robots*, vol. 34, no. 2, pp. 111–127, 2013.
- [25] R. Geraerts and M. H. Overmars, "Creating high-quality paths for motion planning," *International Journal of Robotics Research (IJRR)*, vol. 26, no. 8, pp. 845–863, 2007.
- [26] I. A. Sucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, pp. 72–82, December 2012.
- [27] M. Moll, I. A. Sucan, and L. E. Kavraki, "Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization," *IEEE Robotics & Automation Magazine*, vol. 22, pp. 96–102, September 2015.
- [28] M. Wise, M. Ferguson, D. King, E. Diehr, and D. Dymesich, "Fetch and freight: Standard platforms for service robot applications," in *Workshop on Autonomous Mobile Service Robots*, 2016.