In *Proc. Int. Conf. on Robotics & Automation, 2005*

Hybrid PRM Sampling with a Cost-Sensitive Adaptive Strategy

David Hsu[1,2]
[1]*Department of Computer Science*
National University of Singapore
Singapore
dyhsu@comp.nus.edu.sg

Gildardo Sánchez-Ante[2]
[2]*Computer Science Program*
Singapore MIT Alliance
Singapore
ante@comp.nus.edu.sg

Zheng Sun
Department of Computer Science
Hong Kong Baptist University
Hong Kong S.A.R., China
sunz@comp.hkbu.edu.hk

***Abstract*— A number of advanced sampling strategies have been proposed in recent years to address the narrow passage problem for probabilistic roadmap (PRM) planning. These sampling strategies all have unique strengths, but none of them solves the problem completely. In this paper, we present a general and systematic approach for adaptively combining multiple sampling strategies so that their individual strengths are preserved. We have performed experiments with this approach on robots with up to 12 degrees of freedom in complex 3-D environments. Experiments show that although the performance of individual sampling strategies varies across different environments, the adaptive hybrid sampling strategies constructed with this approach perform consistently well in all environments. Further, we show that, under reasonable assumptions, the adaptive strategies are provably competitive against all individual strategies used.**

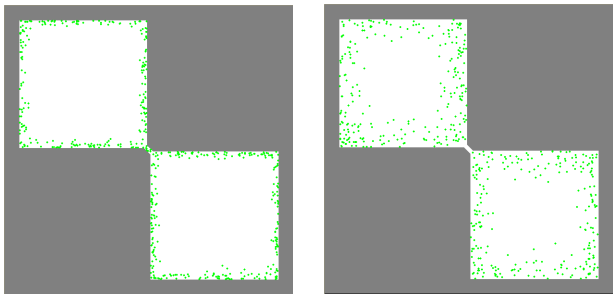
***Index Terms*— robotics, motion planning, randomized algorithms, probabilistic roadmap planners.**

I. Introduction

In recent years, probabilistic roadmap (PRM) planning [10] has emerged as the most successful approach for motion planning of robots with many degrees of freedom (dofs). It has enabled exciting new applications of motion planning in robotics and beyond, *e.g.*, manufacturing, computer animation, and computational biology [11].

The main idea of PRM planning is to *sample* at random a robot's configuration space and capture the connectivity of the space in a graph, called the roadmap. Motion planning is a provably hard computational problem [17]. Despite strong experimental evidence showing the effectiveness of PRM planners, their performance degrades when a robot's configuration space contains narrow passages. This difficulty has long been recognized [1], [8], [10]. Several sampling strategies, also called samplers, have been proposed to address this issue by judiciously picking well-placed points in crucial regions of a robot's configuration space in order to improve the accuracy of PRM planners [1], [3], [5], [6], [7], [8], [12]. These samplers all have unique strengths, but none of them solves the problem completely. More importantly, given a specific motion planning problem, there is no systematic way to determine which sampler is most suitable.

In addition, some of these samplers rely on various parameters to adjust their behaviors for improved performance. For example, the Gaussian sampler [3] selects the standard deviation of the Gaussian sampling distribution. Similarly the randomized bridge builder (RBB) [7] selects the standard deviation of a Gaussian distribution that determines the length of a line segment used to test for narrow passages. For both the Gaussian sampler and RBB, the resulting sampling distributions sometimes differ substantially, depending on the parameter chosen (Fig. 1). Experiments show that by choosing reasonable values for the parameters, these more sophisticated samplers perform significantly better than the uniform sampler used in the classic PRM planner. However, these parameters depend on configuration space properties, which are difficult to obtain in advance. It is thus desirable to set these parameters automatically.



small standard deviation large standard deviation

Fig. 1. Sampling distributions generated by Gaussian samplers with different standard deviations. Shaded areas indicate obstacles. Black dots indicate sampled points. Note, in particular, the difference in the number of samples in the narrow passage.

Motivated by the above considerations, we present a general and systematic approach for adaptively combining multiple samplers. We call it *adaptive hybrid sampling*. It assembles a set of component samplers with complementary strengths and forms a hybrid sampler that identifies the best component samplers automatically. The component samplers may consist of heterogeneous ones such as the different samplers mentioned earlier, or they may consist of homogeneous ones such as the Gausssian samplers with different parameters. The overall performance of the hybrid sampler certainly depends on the component samplers, but our emphasis is to develop a general approach for combining multiple samplers, independent of the specific samplers used. We want to point out that hybrid sampling

cannot make bad component samplers good; instead, it tries to combine component samplers so that their individual strengths are preserved. We demonstrate this experimentally. We also show, through a competitive analysis, that under reasonable assumptions, the hybrid sampler has guaranteed performance, compared with the best component sampler.

The basic idea of our approach is simple. We observe the performance of component samplers and choose the ones with good performance to run more frequently. However, it is important to observe that the computational costs of running various component samplers may differ. For example, the uniform sampler takes only one collision check in a single trial to sample a point from the configuration space and determine whether it should be added to the roadmap; the more sophisticated RBB takes up to three collision checks. Thus, when measuring the performance of component samplers, we must consider the *cost* involved.

In the following, Section II briefly reviews related work. Section III gives an overview of PRM planning. Section IV presents our algorithm for adaptively combining multiple samplers for PRM planning. Section V and VI report experiments with the algorithm and analyze its performance formally. Finally, Section VII concludes with some thoughts on future directions of research.

II. Related Work

Earlier work on PRM planning explored the idea of using multiple samplers or hybrid samplers [4], [5], [7], [9], [10], [14]. Among the general approaches for combining samplers, one possibility is to partition the configuration space into regions and assign a different sampler for each region manually [4] or through supervised learning [14]. Manual assignment is likely not effective, due to the inherent complexity of motion planning [17]. On the other hand, the learning-based assignment requires a *representative* training set, which is difficult to define and construct. Another approach is to combine samplers by weighting their respective sampling distributions [7]. The question is then how to choose the weights effectively. Our earlier work on combining multiple samplers follows a similar approach used here, but employs an adaptive strategy that does not take into account the costs of component samplers and offers no guaranteed performance [9].

The adaptive strategy that we use here is inspired by the work in the on-line learning, in particular, prediction from expert advice [13] and the multi-armed bandit problem [2]. However, in the work on learning, cost is not considered in selecting component strategies.

III. Overview of PRM Planning

A classic multi-query PRM planner computes feasible motions of a robot in a given environment in two phases. In the roadmap construction phase, it uses a sampler with a suitable distribution to pick at random a set of points, called *milestones*, in $\mathcal{F}$, where $\mathcal{F}$ is the collision-free subset of the robot's configuration space. It uses these milestones as nodes to construct a roadmap graph G by adding an edge between every pair of milestones that can be connected by a simple collision-free path, typically, a straight-line segment. After the roadmap has been constructed, multiple queries can be answered efficiently in the query phase. Each query consists of an initial configuration s and a goal configuration g, and asks for a collision-free path connecting s and g. The planner first finds two milestones s' and g' in the roadmap G such that s (g, respectively) and s' (g', respectively) can be connected by a collision-free path, and then searches for a path between s' and g' in G.

The effectiveness of PRM planners depends on two crucial properties of G: *coverage* and *connectivity*. First, the union of the *visibility sets* of all milestones in G should cover a significant portion of $\mathcal{F}$, where the visibility set of a point $p \in \mathcal{F}$ is defined as the set of all points in $\mathcal{F}$ that can be connected to p via a straight-line segment in $\mathcal{F}$. Otherwise, the PRM planner may have difficulty in connecting a given query configuration to an existing milestone. Second, G should capture the connectivity of the underlying free space $\mathcal{F}$ that it represents. Any two milestones in the same connected component of $\mathcal{F}$ should also be connected by a path in G. Without these two properties, a PRM planner may give false negative answers to many queries.

Clearly a key ingredient of PRM planners is the sampler used for generating the milestones. So we focus on sampling in this paper. Other details on the general framework of PRM planning can be found in [10].

IV. Adaptive Hybrid Sampling

Our adaptive hybrid sampler S_{ada} generates milestones for a roadmap G by invoking a set of component samplers S_1, S_2, ..., S_K. Intuitively, to take advantage of the strengths of different component samplers, S_{ada} should invoke more frequently the ones that are more effective in sampling a given configuration space $\mathcal{C}$. However, it is difficult to compute geometrical properties of $\mathcal{C}$ and decide *a priori* which component samplers are more effective. Thus S_{ada} observes the performance of the component samplers and tries to identify the more effective ones automatically in an on-line fashion. Specifically, S_{ada} maintains a probability p_i for each component sampler S_i. In each step, it invokes S_i with probability p_i and then updates p_i according to the performance of S_i. This way, S_{ada} naturally favors component samplers with good performance and invokes them more frequently. A sketch of the PRM planner using S_{ada} is given in Algorithm 1.

To be effective, S_{ada} needs good ways to measure the performance of component sampler and to adapt their probabilities, which we describe in the two subsections next.

A. Measuring the performance of component samplers

One way to evaluate a component sampler S_i is to measure the *usefulness* of the milestones that S_i generates: How much do these milestones improve the quality of the roadmap G? To answer this question, after inserting a new

Algorithm 1 The PRM planner with an adaptive hybrid sampler.

Let p_i be the probability of picking a component sampler S_i. Initialize $p_i = 1/K$, for $i = 1, 2, \ldots, K$.
for $t = 1, 2, \ldots$ **do**
  Pick a component sampler at random according to the probability vector $[p_1, p_2, \ldots, p_K]$.
  Let $S_{i(t)}$ be the sampler chosen. Run $S_{i(t)}$ and generate a new milestone q.
  Insert q into the roadmap G.
  $N_q \leftarrow$ the set of M milestones closest to q among all existing milestones of G within a distance of $D_{\max}$ from q, where M and $D_{\max}$ are fixed constants.
  for all $q' \in N_q$ **do**
    if q' and q lie in different connected components of G **then**
      Check whether there is a collision-free straight-line path between q and q'. If so, insert an edge between q and q' into G,
  Determine how much the new milestone q' improves G, and update $[p_1, p_2, \ldots, p_K]$ accordingly.

milestone q into G and connect it with existing milestones in G, we classify q into one of three types:

1) The milestone q is not connected to any other milestones in G. As a result, G has one more connected component, which contains the single milestone q.
2) The milestone q is connected to milestones all belonging to the same connected component of G. The number of connected components of G remains the same.
3) The milestone q is connected to milestones that originally belong to different connected components of G. As a result, these connected components are merged into a single connected component.

According to the desirable roadmap properties described in Section III, a milestone of Type 3 is clearly useful, as it improves the connectivity of G. A milestone q of Type 1 is also useful, as it improves the coverage of G. Without q, if a query configuration s lies near q, s may not be visible, *i.e.*, connected via a collision-free straight-line segment, to any milestone in G. The PRM planner would then likely give a false negative answer to the query. For a milestone q of either of these two types, we consider it *useful* and assign a reward of 1.

A milestone q of Type 2 could possibly improve the coverage of G, as q's visibility set may cover a subset of $\mathcal{F}$ not visible to other milestones in G. However, if G already contains many milestones, it is more likely that such a milestone q does not improve the coverage of G. Thus we consider a milestone of Type 2 *useless* and assign a reward of 0.

These same considerations were used in the visibility-based PRM planner [16], and were shown to improve the efficiency of PRM planning by rejecting the milestones that are unlikely useful.

Other reasonable criteria of reward assignment can also be used. For example, we may assign a partial reward to a milestone of Type 2 if it has few other milestones nearby. We may also assign a reward greater than 1 to a Type 3 milestone if it is deemed more important to merge together multiple connected components of G. Our adaptive strategy, to be described in the next subsection, is general and does not depend on a particular reward assignment scheme.

B. Adapting preferences on component samplers

The total reward that a component sampler S_i receives reflects its performance. Thus we would like to design an adaptive strategy that dynamically changes the preferences on the component samplers based on the total rewards received. Ideally such a strategy should have the following desirable properties:

- It should favor a component sampler S_i if S_i frequently discovers useful milestones, and punish S_i if S_i repeatedly draws useless milestones.
- It should be robust. It ensures that every component sampler S_i has a reasonable chance of being picked so that the performance of S_i can be evaluated. Also a bad component sampler may draw a few useful milestones by chance. The adaptive strategy should not change the preference drastically as a result of this.
- It should be responsive. The performance of component samplers may change over time. For example, the uniform sampler may work well in the beginning of roadmap construction, when there is a lot of wide open free space to cover and most of the milestones sampled are useful. Specialized samplers for narrow passages may work well towards the end, when useful milestones are rarely found. The adaptive strategy should respond to such changes quickly.
- It should take into account the fact that the costs of generating a milestone may differ among the component samplers.

Our adaptive sampler S_{ada} maintains a weight w_i for component sampler S_i. These weights "record" the past performance of component samplers. Initially set $w_i = 1$ for all i.

Based on the weights, S_{ada} first computes a *cost-insensitive* probability p_i^* for S_i in each step:

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

where $w_i(t)$ is the weight of S_i in step t and γ is a fixed constant. The probability p_i^* is a weighted sum of two components. The first component is proportional to the weight so that a component sampler with good past performance is chosen with high probability. The second component is the same for all component samplers so that every one has a chance of being chosen, ensuring the robustness of S_{ada}.

To take into account the cost variations among the component samplers, the probability p_i of choosing S_i in each step is defined as

$$p_i = \frac{p_i^*/c_i}{\sum_{j=1}^{K} p_j^*/c_j}, \qquad i = 1, 2, \ldots, K, \tag{2}$$

where c_i is the average cost of running S_i to obtain a milestone and inserting it into the roadmap. Consequently a high-cost component sampler has a smaller probability of being chosen.

Now suppose that a component sampler S_i is chosen and receives a reward x_i. For any $S_j, j \neq i$, we assign its reward $x_j = 0$. To update the weights, we first compute for each component sampler the adjusted reward that takes into account its *cost-insensitive* probability:

$$x_i' = x_i/p_i^*, \qquad i = 1, 2, \ldots, K. \tag{3}$$

Next we update the weights

$$w_i(t+1) = w_i(t)\exp(\gamma x_i'/K), \qquad i = 1, 2, \ldots, K. \tag{4}$$

The new weight is the current weight multiplied by a factor that depends exponentially on the reward received. The exponential factors enable the weights to adapt quickly with the samplers' performance changes so that S_{ada} is responsive.

In Section VI, we will show that, under reasonable assumptions, the adaptive sampler S_{ada} is competitive against all individual component samplers, for a given total computation cost. More precisely, if R is the expected total reward of S_{ada} and $\hat{R}$ is the total reward of the best component sampler, we have

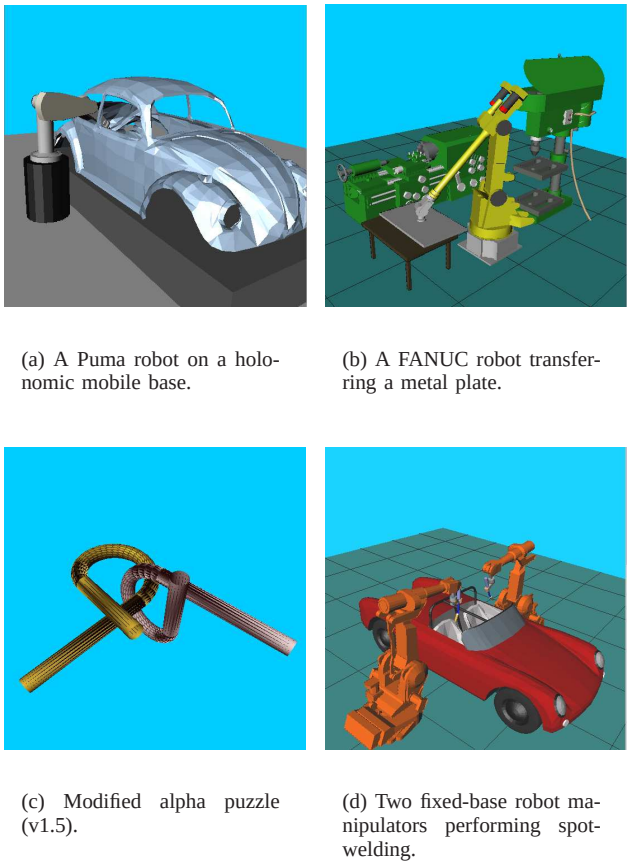
$$\hat{R} - R \leq (\mathrm{e} - 1)\gamma\hat{R} + \frac{K \ln K}{\gamma}. \tag{5}$$

C. Average costs of component samplers

The cost c_i of generating a single milestone by a component sampler S_i can be defined in various ways, depending on our objective.

Often a roadmap is used for processing many queries. So we would like to have a high-quality roadmap and can sometimes afford to spend substantial computation time for roadmap construction. In this case, we simply set $c_i = 1$ for all i, basically ignoring the computational cost in deciding the preferences on component samplers.

On the other hand, if we are interested in not only the quality of the roadmap but also the computation time for roadmap construction, we then define the cost c_i of generating a milestone by S_i to be the number of collision checks invoked by S_i to acquire the milestone and connect it with other existing milestones in the roadmap. To illustrate, suppose that S_{ada} uses two component samplers S_1 and S_2. If both S_1 and S_2 obtain useful milestones with the same frequency, but S_1 uses fewer collision checks than S_2 on the average, S_{ada} will still favor S_1. We use the number of collision checks as a measure of computational cost, because collision checking is the dominant factor in the total computation time for PRM planners.



(a) A Puma robot on a holonomic mobile base.

(b) A FANUC robot transferring a metal plate.

(c) Modified alpha puzzle (v1.5).

(d) Two fixed-base robot manipulators performing spot-welding.

Fig. 2. Test environments.

For many samplers, it is difficult to know in advance the average number of collision checks needed for generating a milestone, but we can estimate it by keeping a history of the number of collision checks used by each component sampler. In our current implementation, we simply use the number of collision checks that S_i uses when it is last chosen to generate a milestone. Initially c_i is set to be 1 for all i, and is updated every time S_i generates a new milestone.

V. Experimental Results

We used four 3-D environments in our experiments (Fig. 2). The manipulator robots used have six dofs. The α-puzzle, shown in Fig. 2(c), can be considered as a free-flying rigid body in 3-D space, also with six dofs.

- The environment in Fig. 2(a) contains several narrow passages through the car's door frame, where the Puma arm can be inserted into the car body. The rest of the space is wide-open. The query used in this case requires the Puma arm to get out from the car body and then be re-inserted into another place. So the robot must go through one narrow passage to a wide-open region and go through another narrow passage.
- For the environment in Fig. 2(b), the FANUC robot has to transfer a metal sheet from a table to a press. The robot's motion near the initial and goal configu-

TABLE I

COMPARING THE PERFORMANCE OF VARIOUS SAMPLERS.

	U	RBB_1	RBB_5	RBB_9	G_1	G_5	G_9	FHS_1	FHS_2	AHS_1	AHS_2
Fig. 2(a)	**1.00**	33.55	49.16	43.89	5.28	5.69	5.78	8.95	9.77	**0.53**	**0.16**
Fig. 2(b)	5.61	2.65	2.27	1.87	**1.00**	1.18	1.51	4.39	2.78	**0.13**	1.06
Fig. 2(c)	17.28	2.21	**1.00**	1.13	3.56	4.31	4.07	7.18	3.56	1.50	1.17
Fig. 2(d)	4.45	6.37	5.19	5.69	1.39	**1.00**	1.58	1.65	1.93	1.15	**0.91**

rations are very constrained. Small changes in almost any dof of the robot may cause a collision. However, the narrow passages here are rather short.

- The environment shown in Fig. 2(c) has been suggested as a benchmark for motion planning problems [1]. It consists of two tubes, each twisted into an alpha shape, and the goal is to separate the intertwined tubes. There are several versions of the problem. The results shown here are for version 1.5, the easiest one. The configuration space corresponding to this environment contains a somewhat long narrow passage.
- Finally, we illustrate the applicability of adaptive hybrid sampling for environments that involve more dofs. The environment in Fig. 2(d) shows two manipulator robots with 6 dofs each and 12 dofs in total.

The configuration spaces for all these environments consists of both narrow passages and relatively open free space. Our purpose is to test how well hybrid samplers adapt in such spaces.

The overall performance of adaptive hybrid samplers depends on two factors: (i) the component samplers and (ii) the strategy for combining them. In our experiments, we used three component samplers of complementary strengths. The uniform sampler (U) provides good coverage of the configuration space. The Gaussian sampler (G) [3] and the Randomized Bridge Builder (RBB) [7] are good for improving the connectivity of roadmaps in narrow passages. Since both the Gaussian sampler and RBB are parametrized, we created several instances of each in our experiments. Other samplers (*e.g.*, [1], [6], [12]) can also be considered. Although the component samplers clearly affect overall performance, making a reasonable, not necessarily the best choice is not difficult, and the main emphasis of this work is to investigate and demonstrate the benefits of combining multiple samplers.

With this goal in mind, we ran experiments with three types of samplers. The first type consists of the individual samplers, U, G_1, G_2, . . ., and RBB_1, RBB_2, . . ., where the subscripts indicate different parameter values. The second type consists of hybrid samplers that combine component samplers with fixed weights. The third type consists of hybrid samplers using the adaptive strategy described in Section IV. We created two such samplers, AHS_1 and AHS_2. AHS_1 uses 10 component samplers: the uniform sampler and nine instances of RBB with different parameters. AHS_2 uses 11 component samplers: the uniform sampler, five instances of RBB and five instances of the Gaussian sampler.

All the experiments were conducted on a Pentium 4 PC with 512 Mb of physical memory.

A. The performance of adaptive hybrid samplers

Table I summarizes the results obtained. For every test environment, we ran the planner until the roadmap being constructed was sufficient to answer the given query. Every test was repeated 20 times, and the results were averaged. The numbers listed represent the relative performance, which is defined as the ratio of the time used by a particular sampler and the time used by the best individual sampler. So the best-performing individual sampler has a ratio of 1.0, and and a ratio smaller than 1.0 indicates performance better than of the best individual sampler. Columns 2–8 of Table I report the relative performance of individual samplers. Due to the space limitation, we show only the individual samplers with the best performance plus some additional ones for comparison. The next two columns report the results for fixed-weight hybrid samplers (FHS), and the last two columns report the results for AHS.

Observe that among the individual samplers, the best performer changes from one environment to another. It is thus difficult for the user to choose a good sampler in advance. For example, consider environment 2(a). Despite the apparent presence of narrow passages, the uniform sampler performs unexpectedly well. The reverse applies to environment 2(b), where most of the space seems empty and the robot has a lot of room to move. Apparently, the regions close to the initial or goal configurations are critical for solving the query, and it takes the uniform sampler more time to succeed.

The fixed-weight hybrid samplers perform better than some of the individual samplers. However, if the choices of the weights are not good, the overhead generated by inefficient samplers can be significant. Consider, for example, environment 2(a). Using the uniform sampler alone leads to good performance. Adding RBB and the Gaussian sampler increases the running time by nearly 9 and 10 times for FHS_1 and FHS_2, respectively.

In contrast, the two adaptive hybrid samplers perform *consistently* well in all environments. They adapt the weights on the component samplers over time so that their performance is always close to that of the best component sampler. They are sometimes even better than the best component sampler running alone, by taking advantage of the complementary strengths of component samplers.

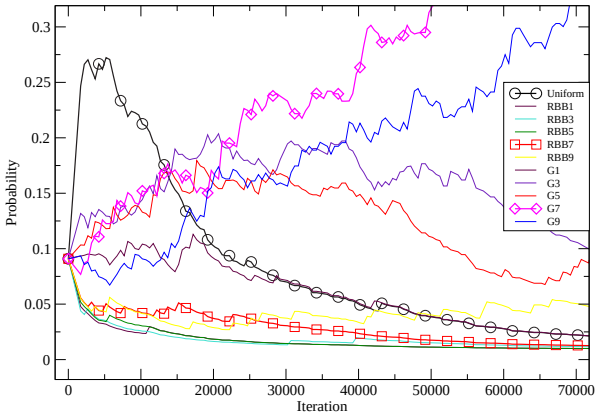


Fig. 3. The change in probabilities for choosing component samplers over time.

B. The behavior of adaptive hybrid samplers

To gain some intuition on the behavior of adaptive hybrid samplers, we recorded the probabilities for picking component samplers at different times. Fig. 3 shows the probability chart for AHS_2 running on environment 2(d). In this chart, we have highlighted three component samplers: the uniform sampler (marked with circles), one RBB (marked with squares), and one Gaussian sampler (marked with diamonds). At the beginning of roadmap construction, the uniform sampler is very useful, and its probability quickly increases, because there is a lot of free space to cover. As more milestones are sampled, the free space becomes well covered, and it is more and more difficult for the uniform sampler to obtain useful milestones. Consequently its probability decreases. Almost the contrary happens with the Gaussian samplers, several of which have probabilities increasing over time. In this environment, the RBBs are not very helpful. The adaptive strategy recognizes this and keeps their probabilities low throughout.

VI. Analysis

The experiments show that the adaptive hybrid sampler S_{ada} performs well. We now analyze the performance of S_{ada} formally through a competitive analysis [15]. Our goal is to show that under reasonable assumptions, the performance of S_{ada} is comparable to that of the best component sampler in terms of the total reward received.

For the sake of analysis, we create a set of *modified samplers* $S'_1, S'_2, \ldots, S'_K$, as well as a *modified adaptive hybrid sampler* S'_{ada} that uses $S'_1, S'_2, \ldots, S'_K$ as component samplers. Our plan is to show first that S'_{ada} is competitive against the best component sampler. We then show that S_{ada} and S'_{ada} are equivalent to obtain the competitive ratio for S_{ada}.

Similar to S_{ada}, S'_{ada} maintains a weight w_i for each component sampler S'_i. In each step, it chooses S'_i with probability p^*_i, specified in (1). If S'_i is not chosen, it receives a reward of 0. Otherwise S'_i runs S_i with probability

$$\frac{1/c_i}{C},$$

where c_i is the cost of running S_i and C is a normalizing factor chosen so that $(1/c_i)/C \leq 1$ for all i. If S_i is run, S'_i receives the reward x_i of S_i and incurs the corresponding cost. If S_i is not run, it receives a reward of 0 and incurs no cost. The weight for each S'_i is then updated according to (4).

Clearly the expected cost c'_i of running S'_i in each step is

$$c'_i = c_i \cdot \frac{1/c_i}{C} + 0 \cdot (1 - \frac{1/c_i}{C}) = \frac{1}{C}$$

So all modified samplers have exactly the same expected cost in each step. We further assume that a new milestone added to the roadmap does not affect the rewards of future milestones significantly. This assumption is valid over a small, finite number of steps. Now S'_{ada} fits exactly into the scenario covered by a multi-armed bandit theorem [2], and it follows that

$$\hat{R}' - R' \leq (\text{e} - 1)\gamma\hat{R}' + \frac{K \ln K}{\gamma}, \tag{6}$$

where R' and $\hat{R}'$ are the expected total rewards of S'_{ada} and its best component sampler, respectively.

Note also that, although S_i and S'_i are not the same, their expected total rewards are. This implies $\hat{R}' = \hat{R}$. Now to get a competitive ratio for S_{ada}, we just need to show that S_{ada} and S'_{ada} are equivalent. For this, let us focus on the how often they run $S_i, i = 1, 2, \ldots, K$, because they can only receive rewards by running S_i and thus adapt the probabilities of choosing their respective component samplers. For S_{ada}, this probability is given by (2). For S'_{ada}, it runs S_i, if it chooses S'_i and S'_i runs S_i, and the probability is $p^*_i \cdot \frac{1}{c_i \cdot C}$. Otherwise, S'_{ada} moves to the next step without doing anything, and the probability is $q = 1 - \sum_{j=1}^{K} \frac{p^*_j}{c_j \cdot C}$. Thus, the probability that S_i is the next sampler that S'_{ada} actually runs is given by

$$\sum_{j=0}^{+\infty} q^j \cdot \frac{p^*_i}{c_i \cdot C} = \frac{1}{1-q} \cdot \frac{p^*_i}{c_i \cdot C} = \frac{p^*_i / c_i}{\sum_{j=1}^{K} p^*_j / c_j},$$

which is exactly same as that in (2). In addition, S_{ada} and S'_{ada} update the weights determine the probabilities of choosing component samplers, only if S_i is run and a reward is received. Thus S_{ada} and S'_{ada} run $S_1, S_2, \ldots, S_K$ with the same probabilities, implying that they are equivalent and $R = R'$. Therefore, we have the following result.

Theorem 1: Suppose that S_{ada} uses K component samplers. Let R and $\hat{R}$ be the expected total reward of S_{ada} and its best component sampler, respectively. For any $K > 0$ and $\gamma \in (0, 1]$,

$$\hat{R} - R \leq (\text{e} - 1)\gamma\hat{R} + \frac{K \ln K}{\gamma}.$$

Assuming that the reward function defined in Section IV-A is a reasonable measure of a roadmap's quality, this result provides the guarantee that the adaptive hybrid sampler will

perform almost as well as the best component sampler, regardless how badly the other component samplers may behave. However, as we have mentioned earlier, this result holds over a small, finite number of steps. It is possible to extend it by periodically resetting the weights of components sampler to 1; however, the bound then becomes weaker and depends on the total number of steps run.

VII. Conclusion and Future Work

This paper presents an adaptive strategy for combining multiple samplers for PRM planning. Experiments show that our adaptive hybrid sampler S_{ada} has *consistently* good performance for both rigid and articulate robots with up to 12 dofs in complex 3-D environments. Furthermore, under reasonable assumptions, S_{ada} is provably competitive against its best component samplers in terms of the total rewards received.

So far, we have been mostly comparing our S_{ada} against individual component samplers. However, some preliminary experiments in Section V indicate that S_{ada} or a modification of it may be competitive against any fixed-weight combination of component samplers. We are currently exploring this possibility from both experimental and theoretical angles.

Acknowledgments

This work was supported in part by NUS under grant R252-000-145-112 and by RGC under grant HKBU2107/04E. We thank Lee Wee Sun from the National University of Singapore for many valuable discussions of the multi-armed bandit problem.

References

[1] N. Amato, O. Bayazit, L. Dale, C. Jones, and D. Vallejo, "OBPRM: An obstacle-based PRM for 3D workspaces," in *Robotics: The Algorithmic Perspective: 1998 Workshop on the Algorithmic Foundations of Robotics*, P. Agarwal *et al.*, Eds. Wellesley, MA: A. K. Peters, 1998, pp. 155–168.

[2] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire, "The nonstochastic multiarmed bandit problem," *SIAM J. Computing*, vol. 32, no. 1, pp. 48–77, 2002.

[3] V. Boor, M. Overmars, and F. van der Stappen, "The Gaussian sampling strategy for probabilistic roadmap planners," in *Proc. IEEE Int. Conf. on Robotics & Automation*, 1999, pp. 1018–1023.

[4] L. Dale and N. Amato, "Probabilistic roadmaps—putting it all together," in *Proc. IEEE Int. Conf. on Robotics & Automation*, 2001, pp. 1940–1947.

[5] M. Foskey, M. Garber, M. Lin, and D. Manocha, "A Voronoi-based hybrid motion planner," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2001, pp. 55–60.

[6] C. Holleman and L. E. Kavraki, "A framework for using the workspace medial axis in PRM planners," in *Proc. IEEE Int. Conf. on Robotics & Automation*, 2000, pp. 1408–1413.

[7] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," in *Proc. IEEE Int. Conf. on Robotics & Automation*, 2003, pp. 4420–4426.

[8] D. Hsu, L. Kavraki, J. Latombe, R. Motwani, and S. Sorkin, "On finding narrow passages with probabilistic roadmap planners," in *Robotics: The Algorithmic Perspective: 1998 Workshop on the Algorithmic Foundations of Robotics*, P. Agarwal *et al.*, Eds. Wellesley, MA: A. K. Peters, 1998, pp. 141–154.

[9] D. Hsu and Z. Sun, "Adaptive hybrid sampling for probabilistic roadmap planning," National University of Singapore, Singapore, Tech. Rep. TRA5/04, May 2004.

[10] L. Kavraki, P. Švestka, J. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration space," *IEEE Trans. on Robotics & Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[11] J. Latombe, "Motion planning: A journey of robots, molecules, digital actors, and other artifacts," *Int. J. Robotics Research*, vol. 18, no. 11, pp. 1119–1128, 1999.

[12] J.-M. Lien, S. Thomas, and N. Amato, "A general framework for sampling on the medial axis of the free space," in *Proc. IEEE Int. Conf. on Robotics & Automation*, 2003, pp. 4439–4444.

[13] N. Littlestone and M. Warmuth, "The weighted majority algorithm," *Information & Computation*, vol. 108, no. 2, pp. 212–261, 1994.

[14] M. Morales, L. Tapia, R. Pearce, S. Rodriguez, and N. Amato, "A machine learning approach for feature-sensitive motion planning," in *Proc. The Sixth Int. Workshop on the Algorithmic Foundations of Robotics*, 2004.

[15] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge University Press, 1995.

[16] C. Nissoux, T. Siméon, and J.-P. Laumond, "Visibility based probabilistic roadmaps," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 1999, pp. 1316–1321.

[17] J. Reif, "Complexity of the mover's problem and generalizations," in *Proc. IEEE Symp. on Foundations of Computer Science*, 1979, pp. 421–427.