THE NATIONAL UNIVERSITY
of SINGAPORE

School of Computing
Lower Kent Ridge Road, Singapore 119260

TRA5/04

Adaptive Hybrid Sampling for Probabilistic Roadmap Planning

David HSU and Zheng SUN

May 2004

Technical Report

Foreword

Adaptive Hybrid Sampling for Probabilistic Roadmap Planning

David Hsu[1] and Zheng Sun[2]

[1] National University of Singapore, Singapore
[2] Hong Kong Baptist University, Kowloon Tong, Hong Kong

Abstract. Several sophisticated sampling strategies have been proposed recently to address the narrow passage problem for probabilistic roadmap (PRM) planning. They all have unique strengths and weaknesses in different environments, but none seems sufficient on its own in general. In this paper, we propose a systematic approach for adaptively combining multiple sampling strategies for PRM planning. Using this approach, we describe three adaptive hybrid sampling strategies. Two are motivated by theoretical results from the computational learning theory. Another one is simple and performs well in practice. We tested them on robots with two to eight degrees of freedom in planar workspaces. In these preliminary tests, the adaptive hybrid sampling strategies showed consistently good performance, compared with fixed-weight hybrid sampling strategies.

1 Introduction

Random sampling has lead to great progress in motion planning of robots with many degrees of freedom (dofs) [1,4–6,8,9,12–14,17,18]. Using random sampling, probabilistic roadmap (PRM) planners reliably solved motion planning problems for multiple robots with 36 or more dofs (see, *e.g.*, [18]), a task that had not been accomplished before by deterministic algorithms. However, it was recognized from early on that narrow passages in a robot's configuration space are a bottleneck for PRM planners [13]. There are several sampling strategies for dealing with narrow passages, including sampling more densely near obstacle boundaries [1,5], retracting to the medial axis [8,10,15,20], or looking for specific patterns of local geometry [11]. They all have unique strengths and weaknesses in different environments, but none seems sufficient on its own in general. Furthermore, given a particular environment, it is not clear how to choose a good sampling strategy. In this paper, we take the first step towards proposing a systematic approach for adaptively combining multiple sampling strategies and constructing hybrid sampling strategies for PRM planning.

Some earlier work on PRM planning used multiple sampling strategies. One approach is to partition the configuration space into possibly overlapping regions and use a different sampling strategy for each region [7]. However, partitioning a high-dimensional space into regions and maintaining them are difficult. Another approach is to combine sampling strategies by weighting [11]. Suppose that two samplers have distributions π_1 and π_2, respectively. A weighted hybrid sampler then has the distribution $(1-w)\pi_1+w\pi_2$, where $w \in [0,1]$ is the weight. For example, π_1 may sample the configuration space uniformly, and π_2 peaks in narrow passages. By combining the two distributions, the hybrid sampler may cover the entire configuration space well. Unfortunately it is difficult to choose the best value for w. It has been observed that setting $w = 0.5$ often gives good performance experimentally, but clearly there are environments where more extreme values of w give better performance. Furthermore, if there are n component samplers with n much larger than 2 and they are all weighted equally, the performance likely deteriorates, because the best component sampler gets only a weight of $1/n$ and does not run frequently enough.

Instead of trying to set good values for the weights *a priori*, we propose to set them adaptively. Our basic idea is simple. We observe the performance of component samplers. In each iteration, we adjust the weights of component samplers based on their past performance and pick samplers to run with probabilities that depend on the weights. Over the time, samplers with good performance have higher weights and run more frequently. If we formulate the sampling problem this way, results from the computational learning theory indicate that certain algorithms for updating the weights are provably good [2,3,16]. More precisely, they are competitive against the "optimal" one. We describe two adaptive sampling strategies using these algorithms. However, these algorithms consider various worst-case scenarios, some of which are not important in our problem. We show through experiments that a simple heuristic algorithm actually performs slightly better than these two.

In the rest of the paper, Section 2 gives motivation of our approach by reviewing relevant results from the learning theory. Section 3 formulates the adaptive hybrid sampling approach. Section 4 describes three algorithms for updating the weights. Section 5 shows how to use our adaptive hybrid sampling approach to combine the bridge test [11] and uniform sampling. Section 6 reports experimental results. Section 7 discusses some alternatives to the choices made in our approach. Section 8 summarizes the results and points out future research directions.

2 Preliminaries

Predicating from expert advice is studied extensively in the learning theory [3]. In a model problem, we would like to predict the outcomes of a sequence of trials, *e.g.*, horse races. There is a panel of n experts, each making an independent prediction. In every trial, we make a predication based on the n expert predictions. The true outcome is then revealed to us.

If we knew that expert i is the most successful predicator, then following the prediction of i is our best choice, but if we do not know the best expert, what shall we do? Algorithm 1 seems a fairly intuitive solution [16]. We assign a weight to each expert. In every trial, we choose to follow the prediction of an expert with probability proportional to the expert's weight. When the true outcome is revealed, all the experts making correct predictions are rewarded: their weights are increased by a constant factor $1 + \eta$. Surprisingly, this simple algorithm is guaranteed to do almost as well as the best expert. It has been shown that in any sequence of trials,

Algorithm 1 Randomized Weighted Majority.

```
1: Let $w_i(t)$ be the weight of expert $i$ at $t$th trial. Initialize $w_i(0) = 1$ for $i = 1, 2, \dots, n$.
2: for $t = 1, 2, \dots$ do
3:     Set the probability
           $p_i = \frac{w_i(t)}{\sum_{j=1}^{n} w_j(t)}, \qquad i = 1, 2, \dots, n.$     (1)
4:     Given the experts' predictions $x_i, i = 1, 2, \dots, n$, output $x_i$ with probability $p_i$.
5:     When the outcome is revealed, set
           $w_i(t+1) = \begin{cases} w_i(t)(1+\eta) & \text{if expert } i\text{'s prediction is correct,} \\ w_i(t) & \text{otherwise,} \end{cases} \quad i = 1, 2, \dots, n,$     (2)
       where $\eta$ is a small positive constant.
```

the expected number of correct predications R made by the Randomized Weighted Majority algorithm satisfies

$$R \geq (1 - \frac{\eta}{2})\hat{R} - \frac{\ln n}{\eta},$$

where $\hat{R}$ the number of correct predictions made by the best expert [16]. For $\eta = 0.5$, we have $R \geq 0.75\hat{R} - 2 \ln n$.

If we treat each sampler as an expert, it seems that the Randomized Weighted Majority algorithm may be applicable to hybrid sampling. This is almost true, after some modifications. First, we must note an important difference. In the prediction problem, every expert makes a prediction for a trial, and the weights of all experts are updated, if necessary, when the outcome is revealed. In the sampling problem, in each iteration, we choose a sampler to pick a point at random, and then evaluate the performance of this sampler according the point chosen. However, we know nothing about the performance of other samplers in this iteration. A good sampler may be "starved" because it is unlucky and never gets a chance to show its performance. This difference leads us to another model problem in the learning theory, the multi-armed bandit problem [2].

In the bandit problem, a gambler chooses to play one of n slot machines, each with a different, unknown reward. The rewards may change over time. The gambler's objective is to maximize the total reward over a sequence of plays.

3 Adaptive Hybrid Sampling

A classic multi-query PRM planner proceeds in two stages [13]. The first stage is pre-computation. The planner samples the configuration space $\mathcal{C}$ at random and constructs a roadmap graph G that captures the connectivity of $\mathcal{C}$. The nodes of G are sampled collision-free points from $\mathcal{C}$ and are called *milestones*. There is an edge between two milestones if they can be connected via simple, collision-free paths, typically, straight-line segments. The second stage is query processing. The planner searches G for a collision-free path between two given query configurations.

In this paper, we follow this general framework, but address only the first stage and focus on combining multiple strategies for sampling $\mathcal{C}$. Methods for the second stage are well-known [1,13].

Assume that our adaptive PRM planner is given n samplers, preferably, of complementary strengths. A hybrid sampler that combines these component samplers by weighting has the distribution

$$\pi = \sum_{i=1}^{n} p_i \pi_i,$$

where π_i is the distribution of component sampler i and p_i is probability of picking sampler i. Choosing the best values for the weights in advance seems difficult. So let us turn this static problem into a dynamic one.

At each time step t, we pick a sampler i with probability $p_i(t)$. Using this sampler, we pick a new milestone q at random from $\mathcal{C}$ and connect q with existing milestones nearby, if possible. Finally, we adjust the probabilities $p_i(t), i = 1, 2, \ldots, n$, and continue to the next step. Choosing the values for $p_i(t)$ is potentially easier, because each time step involves only local considerations of the relative merits of component samplers. At the end of T steps, the hybrid sampler has the distribution

$$\pi' = \frac{1}{T}\sum_{t=1}^{T} \pi'(t) = \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{n} p_i(t)\pi_i = \sum_{i=1}^{n}\left(\frac{1}{T}\sum_{t=1}^{T} p_i(t)\right)\pi_i.$$

This is basically the average of sampling distributions in T steps, and the weight of sampler i is $(1/T)\sum_{t=1}^{T} p_i(t)$. We hope that π' is close to π with the optimal weights.

To update the probabilities $p_1, p_2, \ldots, p_n$, we maintain a weight w_i for every component sampler i and adjust w_i by assigning a reward r based on the new milestone obtained. Our objective is to build a small roadmap that captures the connectivity of the free space well. A good roadmap G has two properties: *coverage* and *connectivity*. Let us say that two points are *visible* to each other, if they can be connected via a straight-line path that lie entirely in the free space $\mathcal{F}$. Then coverage means that the union of the visibility sets of milestones in G covers a significant portion of $\mathcal{F}$. So for any given (query) configuration $p \in \mathcal{F}$, there is a straight-line path between p and a milestone in G with high probability. Furthermore, G should capture the connectivity of the underlying free space $\mathcal{F}$ that it represents: there is a path in G between two milestones q and q', if and only if q and q' lie in the same connected component of $\mathcal{F}$. We evaluate the new milestone and assign a reward r according to these two criteria. After connecting q with other existing milestones in G, we have three possibilities:

- The new milestone q cannot be connected to any other existing milestones in G. This implies that q does not lie in the visibility set of any other milestones in G and likely contributes to the coverage of G. So we set the reward $r = 1$.
- The new milestone q is connected to milestones in two or more connected components of G. In this case, the addition of q merges two connected components of G and improves its connectivity. We also set $r = 1$.
- The new milestone q is connected to milestones in a single connected component of G. The milestone q clearly does not improve the connectivity of G, and it may or may not improve the coverage significantly. We assign a reward of 0.

Similar considerations were used in [17]. After assigning the reward r, we adjust the weights $w_1, w_2, \ldots w_n$ and the probabilities $p_1, p_2, \ldots p_n$ accordingly.

A sketch of our planner is shown below.

Algorithm 2 Adaptive PRM planner.

```
1: Let $w_i$ be the weight of sampler $i$. Initialize $w_i = 1$ for $i = 1, 2, \ldots, n$.
2: for $t = 1, 2, \ldots$ do
3:     Pick a sampler $i$ with probability $p_i$ that depends on $w_i$.
4:     Suppose that sampler $i_t$ is picked. Apply this sampler and pick a new milestone $q$.
5:     Sort the existing milestones in $G$ according to their distance to $q$ using a suitable
       distance metric.
6:     For every milestone $q'$ among the $K$ closest to $q$, add an edge between $q$ and $q'$, if $q$ and
       $q'$ are within a pre-specified distance and they can be connected with a collision-free
       straight-line path. $K$ is a fixed constant.
7:     Let $V_q$ be the set of milestones that are connected to $q$. Set the reward
       $$r = \begin{cases} 1 & \text{if } V_q \text{ is empty,} \\ 0 & \text{if all the milestones in } V_q \text{ lie in one connected component of } G, \\ 1 & \text{if the milestones in } V_q \text{ lie in two or more connected components of } G. \end{cases}$$
8:     Adjust the weights based on the reward $r$.
```

By maximizing the total reward over the iterations, this algorithm tries to build a roadmap that has a small number of milestones and captures the connectivity of the free space well. A small roadmap reduces storage requirement and speeds up the query time in the second stage of PRM planning. In addition, a small roadmap often leads to fast pre-computation time as well, as we will see in Section 6.

Since an adaptive hybrid sampling strategy combines a set of component samplers, we cannot expect it do well if all the component samplers are bad. Our goal is therefore to design an adaptive strategy that compares favorably with the *best* fixed-weight hybrid sampling strategy in a wide variety of environments.

4 Updating the Weights

To complete our description of adaptive sampling, we now specify how the probability p_i of choosing sampler i depends on the weight w_i and how w_i is updated. There are several options. While some offer theoretical guarantees and others perform well experimentally, they all share some basic considerations:

Reward and penalty In essence, the weights reflect the past performance of samplers. We try to identify good samplers through their weights. The weight of a sampler is increased if it receives high rewards *frequently* by obtaining useful milestones. The weight is decreased otherwise. A similar idea is used in Algorithm 1 for the prediction problem.

Robustness Sometimes a good sampler may have a small weight, due to random variations. It is then seldom picked and can never have its weight increased. In other words, the weight of a sampler remains small, not because it performs badly, but because it never has a chance to show its performance. We must prevent this from happening by ensuring that every sampler has a reasonable chance of being picked. This is similar to the classic trade-off between exploration and exploitation in reinforcement learning [19].

Responsiveness The rewards for the samplers may change over time. For example, uniform sampling may work well in the beginning, when there is a lot of wide open space. Once such space is covered, the performance of uniform sampling deteriorates. The weights must respond to these changes. Therefore the recent performance history is more important than that a long time ago.

In the following subsections, we describe three methods, AS1, AS2, and AS3, for updating the weights. The first two are derived from algorithms originally designed for the multi-armed bandit problem [2]. The last one is simple, but works well in practice.

4.1 AS1: Competing Against the Best Fixed-Weight Strategy from a Pool

Suppose that there are n component samplers. We first construct a pool of m fixed-weight hybrid samplers (FHS). Each FHS has an associated constant weight vector $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_n)$, where the *i*th element β_i represents the probability of sampling component sampler i. Let $\boldsymbol{\beta}^j$ denote the weight vector of the jth FHS. We maintain a weight w_j for each FHS j.

Now in iteration t, we choose component sampler i with probability

$$p_i = (1-\gamma)\sum_{j=1}^{m}\frac{w_j(t)\beta_i^j}{\sum_{k=1}^{m} w_k(t)} + \gamma\frac{1}{n}, \qquad i = 1, 2, \ldots, n, \tag{3}$$

where $w_j(t)$ is the weight of the *j*th FHS in iteration t and $\gamma \in (0, 1]$ is a fixed constant. The probability p_i is a weighted sum of two components. The first component is proportional to the weight, similar to that used in Algorithm 1. A sampler with good past performance is chosen with high probability. Here the weight w_j is multiplied by β_i^j, because our expert samplers here are not the component samplers, but the fixed-weight hybrid samplers. The second component of p_i is basically uniform sampling. It is the same for all component samplers so that every sampler has a chance of being chosen, in order to ensure robustness.

Suppose that component sampler i_t is picked in iteration t and receives a reward r. We scale the reward by a factor of $1/p_{i_t}$ to take into account how frequently sampler i_t is chosen. If a component sampler is not picked, it receives no reward. In formula, we have

$$r_i = \begin{cases} r/p_i & \text{if } i = i_t, \\ 0 & \text{otherwise}, \end{cases} \qquad i = 1, 2, \ldots, n.$$

Next we set the reward of FHS j by taking a linear combination of $r_1, r_2, \ldots, r_n$ using the weight vector $\boldsymbol{\beta}^j$:

$$s_j(t) = \sum_{i=1}^{n} \beta_i^j r_i(t), \qquad j = 1, 2, \ldots, m.$$

Finally we update the weight of every FHS:

$$w_j(t+1) = w_j(t)\exp(\gamma s_j(t)/n), \qquad j = 1, 2, \ldots, m. \tag{4}$$

The weight is multiplied by an factor that depends exponentially on the reward received. This is again similar to that used in Algorithm 1. The exponential factors ensure that the weights change quickly with the samplers' performance so that the adaptive algorithm is responsive.

If this update method is applied to the multi-armed bandit problem described in Section 2, one can show that the adaptive algorithm is competitive against the best fixed-weight algorithm [2]. More precisely, if R and $\hat{R}$ are the expected total reward of the adaptive algorithm and the best fixed-weight algorithm respectively, then

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

This result applies only partially to our sampling problem, because in the bandit problem, although the rewards may vary over time, they are assigned in advance. In our problem, every new milestone added to the roadmap may affect the rewards of future milestones. To apply this result, we have to assume that the effect of adding a milestone is small. This assumption is valid over a small, finite number of iterations.

Now how do we construct a pool of fixed-weight hybrid samplers? One way is to discretize the weights and enumerate all the possibilities. The size of the pool then depends exponentially on n, the number of component samplers, and we must maintain a large number of weights. This is feasible only if n is very small.

4.2 AS2: Competing Against Arbitrary Strategies

The difficulty of constructing a pool of hybrid samplers leads us to the second method for updating the weights. It maintains a weight w_i for each component sampler i and chooses a sampler i in iteration t with probability

$$p_i = (1-\gamma)\frac{w_i(t)}{\sum_{j=1}^{n} w_j(t)} + \gamma\frac{1}{n}, \qquad i = 1, 2, \ldots, n. \tag{6}$$

To update the weights, we first set the reward for each component sampler:

$$r_i = \begin{cases} r/p_i & \text{if } i = i_t, \\ 0 & \text{otherwise,} \end{cases} \qquad i = 1, 2, \ldots, n,$$

as in the previous section. Next we set the new weights

$$w_i(t+1) = w_i(t)\exp(\gamma r_i/n) + \frac{\mathrm{e}\alpha}{n}\sum_{j=1}^{n} w_j(t), \qquad i = 1, 2, \ldots, n, \tag{7}$$

where α is a small positive constant. The new weight consists of two terms. The first term is same as that in (4). The second term is a fraction of the average weight, so that no individual weight becomes too small relative to the others. This is again to ensure robustness: the probability of picking an under-performing sampler quickly rises, if its performance improves.

It can be shown that if applied to the multi-armed bandit problem, this update method is competitive against any arbitrary sequence of choices of component samplers [2]. The bound has a form similar to (5), but depends on the number of times that the optimal strategy switches the samplers. Again, in our sampling problem, this bound is only applicable over a small, finite number of iterations.

4.3 AS3: Equal-Weight with Variable-Length History

The weights of component samplers reflect their past performance. A simple way for tracking performance is to maintain an observation queue for each component sampler. An element of the queue records the reward that the sampler receives when chosen to pick a milestone. The average reward in the queue then gives a good estimate on the sampler's past performance.

The question now is how long the queue should be. If the queue is very long, the estimate may be more reliable. However, when the samplers' performance changes, the weights change slowly due to the long queue length, and the adaptive sampler may not be responsive enough. On the other hand, if the queue is too short, the estimate may be unreliable, and we cannot differentiate the samplers' performance due to random variations. Consider, for example, two samplers. One receives a reward of 1 out of 10 milestones that it picks. The other receives a reward of 1 out of 100 milestones that it picks. If the queue length is three, then both of them may have received 0 rewards for the last three milestones and appear to have equal performance. Therefore, ideally we would like the queue to be short enough to reflect the samplers' recent performance and long enough to be robust against random variations.

To achieve this goal, we use variable queue lengths. Every time that a sampler is picked, we append its reward to the end of the queue. Hence the queue length increases by 1. If the reward is 1, we reduce the queue length by a half and keep only the recent elements in the queue, because at this moment, the queue likely contains enough information for reliable estimation of the performance. If the reward is 0, we do not change the queue length so that the queue is long enough to contain sufficient information for estimating the performance.

For each component sampler i, we maintain a weight w_i and a queue of length L_i. We initialize the queues by setting $L_i = 1$ for all i and inserting a reward of 1 into each queue. In iteration t, we pick sampler i with probability

$$p_i = \frac{w_i(t)}{\sum_{j=1}^{n} w_j(t)}, \qquad i = 1, 2, \ldots, n. \tag{8}$$

Suppose that sampler i_t is picked and receives a reward r. We insert the reward into i_t's queue and update its queue length:

$$L_{i_t}(t+1) = \begin{cases} (L_{i_t}(t) + 1)/2 & \text{if } r = 1, \\ L_{i_t}(t) + 1 & \text{otherwise.} \end{cases} \tag{9}$$

Finally we update the weights. The weight w_i is simply the average of rewards in sampler i's queue:

$$w_i(t+1) = \frac{\sum_{j=1}^{L_i(t+1)} r_j^i}{L_i(t+1)}, \qquad i = 1, 2, \ldots, n, \tag{10}$$

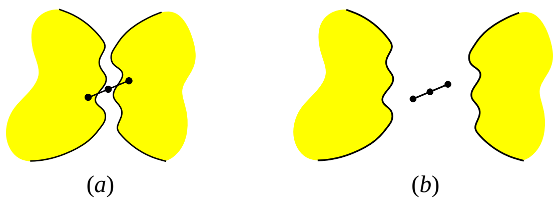

Fig. 1. Building short bridges is much easier in narrow passages than in wide-open free space.

where r_i^j is the *j*th element in sample i's current queue.

Note that the reward is either 1 or 0 according to our definition. With proper initialization, the update rule (9) guarantees that every queue contains at least one strictly positive reward. In addition, if a component sampler receives a reward of 1 for every k milestones that it picks, the queue length varies between k and $2k$, large enough to give a reasonable estimation of the sampler's performance.

5 The hybrid bridge test

To evaluate our adaptive hybrid sampling approach, we use it for combining the bridge test [11] and uniform sampling.

The bridge test is a specialized sampler designed to boost the sampling density inside narrow passages. It is based on the following observation. A narrow passage in a *d*-dimensional configuration space has at least one restricted direction v such that a small perturbation of the robot's configuration along v results in collision of the robot with obstacles. Therefore, for a collision-free configuration q in a narrow passage, it is easy to sample at random a short line segment s through p such that the endpoints of s lie in obstacles in $\mathcal{C}$ (Figure 1*a*). The line segment s is called a *bridge*, because it resembles a bridge across the narrow passage. We say that a point $p \in \mathcal{F}$ passes the bridge test, if we succeed in obtaining such a segment s through p. Clearly building *short* bridges is much easier in narrow passages than in wide-open free space (Figure 1). By favoring short bridges over longer ones, we increase the chance of accepting points in narrow passages. To sample a new milestone using the bridge test, we pick a line segment s from $\mathcal{C}$ at random by choosing its endpoints and determine whether s passes the bridge test. If so, we insert the midpoint of s into the roadmap G as a new milestone.

While being very effective in boosting the sampling density inside narrow passages, the bridge test severely reduces the sampling density in wide-open collision-free regions. This may be undesirable, because an adequate number of nodes are needed in the roadmap to cover the entire free space. The difficulty encountered by the bridge test can be overcome by uniform sampling, which tends to place many samples in wide-open free space. Therefore we use the adaptive hybrid sampling approach to combine the bridge test and the uniform sampling.

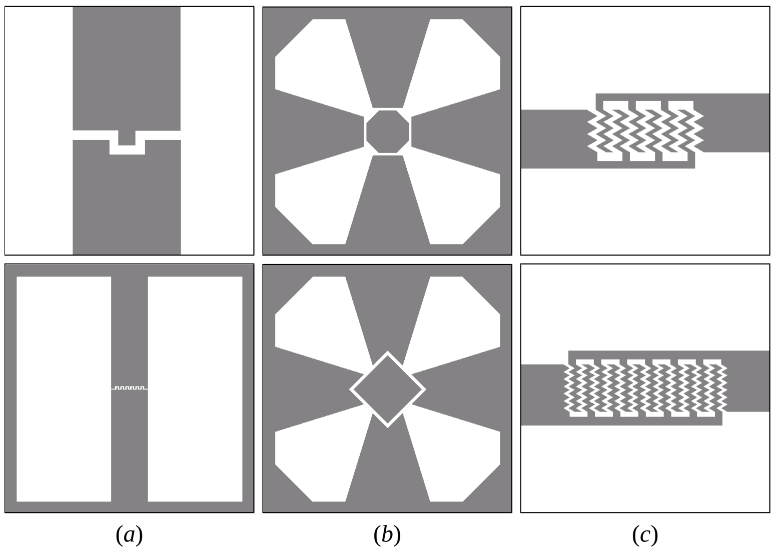

Fig. 2. Test environments with point robots.

6 Experiments

We implemented in Java the adaptive PRM planners with the three methods for updating the weights and conducted preliminary tests. In these tests, the uniform sampler and the bridge test are used as component samplers. The test environments consist of robots with two to eight dofs in planar workspaces. These environments differ in the weights needed for optimally combining the component samplers, so that we can compare the performance of the adaptive hybrid sampling strategies and the best fixed-weight hybrid sampling strategy.

We now briefly describe the test environments below.

- Figure 2: There are three sets of test environments here, all using point robots. In Figure 2*a*, the workspaces contain two large chambers joined with a narrow passage with multiple turns. In Figure 2*b*, the workspaces contain four chambers joined with narrow passages. To go from one chamber to another, the robot must pass through one of the narrow passages. For Figure 2*c*, we have four workspaces of this type with narrow passages of different lengths. The figures show the simplest and the most complex one.
- Figure 3*a*: We have a rigid-body robot that need to go from one large empty chamber to anther by passing through a opening in the lower middle of the figure. We have six workspaces of this type with openings of different sizes. If the opening is wide, the robot can go through easily. If the opening is small, the robot has translate and rotate simultaneously in order to pass through.
- Figure 3*b*: In this environment, a rigid segment needs to enter a narrow corridor, reorient in a small circular room, and exits another narrow corridor.

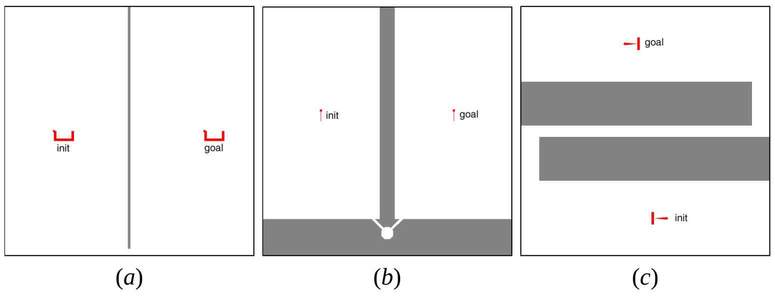

Fig. 3. Test environments with rigid or low-dof robots.

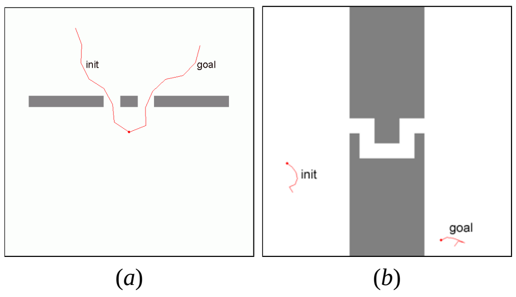

Fig. 4. Test environments with articulated robots.

- Figure 3*c*: Here we have a T-shaped robot with two parts, a “torso” and a “shoulder”, connected with a joint. The workspace contains a long and narrow corridor with two turns. Inside the corridor, the robot must extend itself by aligning its shoulder with the torso to avoid colliding with the walls of the corridor. However, to make a turn in the corridor, the robot has to rotate the shoulder so that it becomes almost perpendicular to the torso. Otherwise the robot cannot turn around the corner.
- Figure 4*a*: This example contains a seven-dof articulated robot with a fixed base. At both the initial and goal configurations, the robot is trapped inside narrow openings and must execute difficult maneuvers in order to find a path.
- Figure 4*b*: The environment contains a relatively long corridor with two turns. So each milestone in the corridor has low visibility and covers only a small portion of the free space. The robot is an articulated robot with six links and a mobile base, eight dofs in total.

6.1 Comparing Adaptive Hybrid Sampling and Fixed-Weight Hybrid Sampling

We tested the three adaptive sampling strategies along with seven fixed-weight sampling strategies for comparison. For every environment, we specified one or more queries and ran a planner until the roadmap being constructed was sufficient for answering the specified queries. Every test was repeated ten times, and the results were averaged.

Table 1 shows the performance statistics for the various various sampling strategies listed by the columns. The second row of the table shows the weight used by the fixed-weight sampling strategies for combining component samplers, where a weight w means that the probability of picking the uniform sampler is w and the probability of picking the bridge test is $1 - w$. There are four rows corresponding to every environment. N_{mil} and N_{col} are respectively the number of milestones and the number of collision checks used for constructing a roadmap. P_{mil} is the relative performance, defined as the ratio of N_{mil} over the minimum number of milestones used by all the sampling strategies. A sampler with $P_{\mathrm{mil}} = 1$ has the best performance, using N_{mil} as the criterion. P_{col} is similarly defined using N_{col}.

According to the relative performance P_{mil}, most fixed-weight sampling strategies tend to perform well in some environment and not in others. The adaptive sampling strategies perform consistently well in almost all the environments. Their performance is usually not more than 20% worse than that of the *best* fixed-weight sampling strategy and is never more than 50% worse. In comparison, a fixed-weight algorithm may be several times worse. In several tests, including the two with seven- and eight-dof articulated robots, adaptive sampling strategies have the best performance among all sampling strategies tested.

To compare the three adaptive sampling strategies, we calculated the average of their relative performance P_{mil} over all the tests. The results are shown in the last row of Table 1. The results for the fixed-weight sampling strategies are also shown for reference. In general, the adaptive sampling strategies perform better than any of the fixed-weight sampling strategies. It is interesting to observe that AS3, the simplest adaptive strategy, performs quite well. In fact, it is slight better than the other two more sophisticated one. AS2 performs the worst among the adaptive strategies. We suspect that since AS2 tries to control the differences in weights among the component samplers, it is not aggressive enough in exploiting the one with best performance. More tests are needed to confirm this.

Among the fixed-weight sampling strategies, FHS 1/2 performs the best and is comparable to AS2. This is not surprising, because many test environments here contain both large collision-free regions and narrow passages and FHS 1/2 gives equal weights to the uniform sampler and the bridge test.

We want to emphasize that the statistics in the last row of Table 1 depends on the chosen test environments. If we add a few environments with no narrow passages, the exact numbers will change. However, we believe that our general observations regarding the adaptive sampling strategies will remain valid.

In our adaptive sampling approach, we have chosen a reward function that minimizes the number of milestones in the roadmap. A small roadmap often leads to fast computation time as well. This is confirmed by our experiments. Although our reward function does not take into account the computational cost explicitly, the performance of adaptive sampling strategies are often quite competitive with the best fixed-weight sampling strategy, according to the relative performance P_{col}. We use the number of collision checks as a measure of computational cost, because collision checks dominate in the running time of PRM planners. It allows us to run many tests simultaneously on different machines without concerning about the variations due to machines and system load.

Table 1: The performance of various hybrid sampling strategies.

Env.		FHS	FHS	FHS	FHS	FHS	FHS	FHS	AS3	AS2	AS1
(Fig.)		1/16	1/8	1/4	1/2	3/4	7/8	15/16			
2*a*	N_{mil}	1113	553	292	186	152	175	252	179	188	164
	P_{mil}	7.35	3.65	1.93	1.23	1.00	1.15	1.66	1.18	1.24	1.08
	N_{col}	167485	80364	39348	20497	12208	10370	11078	17857	20838	15554
	P_{col}	16.15	7.75	3.79	1.98	1.18	1.00	1.07	1.72	2.01	1.50
	N_{mil}	4302	2965	2877	3113	6029	12056	20086	2734	3127	2866
	P_{mil}	1.57	1.08	1.05	1.14	2.21	4.41	7.35	1.00	1.14	1.05
	N_{col}	1517858	1011482	873142	685624	692915	696258	612797	717057	687636	727879
	P_{col}	2.48	1.65	1.42	1.12	1.13	1.14	1.00	1.17	1.12	1.19
2*b*	N_{mil}	669	353	187	101	137	207	309	105	100	108
	P_{mil}	6.70	3.53	1.88	1.01	1.37	2.08	3.09	1.05	1.00	1.08
	N_{col}	103596	53656	27162	11865	10873	11152	11927	11947	11639	12703
	P_{col}	9.53	4.93	2.50	1.09	1.00	1.03	1.10	1.10	1.07	1.17
	N_{mil}	645	352	193	152	230	343	568	160	159	159
	P_{mil}	4.24	2.31	1.27	1.00	1.51	2.26	3.73	1.05	1.05	1.05
	N_{col}	85138	47511	26010	17450	17792	18768	21731	17587	18160	18176
	P_{col}	4.88	2.72	1.49	1.00	1.02	1.08	1.25	1.01	1.04	1.04
2*c*	N_{mil}	1819	927	785	1090	1818	2584	3278	918	1098	878
	P_{mil}	2.32	1.18	1.00	1.39	2.32	3.29	4.17	1.17	1.40	1.12
	N_{col}	347616	277882	267565	299200	351005	358380	342660	282575	300651	277935
	P_{col}	1.30	1.04	1.00	1.12	1.31	1.34	1.28	1.06	1.12	1.04
	N_{mil}	1675	1195	1262	1830	2977	4451	6828	1457	1821	1393
	P_{mil}	1.40	1.00	1.06	1.53	2.49	3.72	5.71	1.22	1.52	1.17
	N_{col}	483457	458349	465805	536236	610222	606826	608901	486440	531548	473291
	P_{col}	1.05	1.00	1.02	1.17	1.33	1.32	1.33	1.06	1.16	1.03
	N_{mil}	1730	1472	1717	2461	4606	7263	9790	1775	2452	1831
	P_{mil}	1.18	1.00	1.17	1.67	3.13	4.93	6.65	1.21	1.67	1.24
	N_{col}	597517	584873	624850	729751	801062	787507	754726	633707	730754	641777
	P_{col}	1.02	1.00	1.07	1.25	1.37	1.35	1.29	1.08	1.25	1.10
	N_{mil}	2250	2334	2620	3862	6529	10755	14351	2847	3868	2776
	P_{mil}	1.00	1.04	1.16	1.72	2.90	4.78	6.38	1.27	1.72	1.23
	N_{col}	760372	790932	855248	1007718	1035480	996845	939488	881965	1005656	877832
	P_{col}	1.00	1.04	1.12	1.33	1.36	1.31	1.24	1.16	1.32	1.15
3*a*	N_{mil}	298	227	192	262	520	942	1773	284	252	301
	P_{mil}	1.55	1.18	1.00	1.36	2.71	4.90	9.23	1.48	1.31	1.57
	N_{col}	129789	94001	73004	76797	101187	140429	228454	78212	75842	81124
	P_{col}	1.78	1.29	1.00	1.05	1.39	1.92	3.13	1.07	1.04	1.11
	N_{mil}	416	288	174	135	207	359	555	144	131	149
	P_{mil}	3.17	2.19	1.33	1.03	1.58	2.73	4.23	1.10	1.00	1.13
	N_{col}	170584	112289	61895	36990	38266	47626	60873	35347	35759	40334
	P_{col}	4.83	3.18	1.75	1.05	1.08	1.35	1.72	1.00	1.01	1.14
	N_{mil}	387	251	163	102	118	183	307	97	103	104
	P_{mil}	3.98	2.58	1.68	1.05	1.22	1.88	3.16	1.00	1.06	1.07
	N_{col}	146300	90293	54056	26187	21038	23255	31127	23970	26805	26472
	P_{col}	6.95	4.29	2.57	1.24	1.00	1.11	1.48	1.14	1.27	1.26
	N_{mil}	456	306	174	101	85	127	178	102	97	100
	P_{mil}	5.39	3.61	2.05	1.19	1.00	1.50	2.10	1.20	1.15	1.18

Table 1 cont'd

	N_{col}	170438	109733	56449	25873	14797	17027	18821	25498	24477	26914
	P_{col}	11.52	7.42	3.82	1.75	1.00	1.15	1.27	1.72	1.65	1.82
	N_{mil}	503	321	185	106	89	106	141	107	103	109
	P_{mil}	5.65	3.60	2.08	1.19	1.00	1.19	1.58	1.20	1.16	1.23
	N_{col}	197556	118420	61478	26727	15889	14454	15454	26596	26090	29119
	P_{col}	13.67	8.19	4.25	1.85	1.10	1.00	1.07	1.84	1.80	2.01
	N_{mil}	376	302	215	146	123	108	101	141	144	154
	P_{mil}	3.72	2.99	2.13	1.45	1.22	1.07	1.00	1.40	1.42	1.52
	N_{col}	215164	162173	99902	48856	26070	16547	12410	42629	46277	52322
	P_{col}	17.34	13.07	8.05	3.94	2.10	1.33	1.00	3.44	3.73	4.22
3*b*	N_{mil}	466	499	582	865	1712	3284	5507	686	858	709
	P_{mil}	1.00	1.07	1.25	1.86	3.67	7.04	11.81	1.47	1.84	1.52
	N_{col}	233219	238416	251035	289033	368119	407976	406592	260165	287502	264616
	P_{col}	1.00	1.02	1.08	1.24	1.58	1.75	1.74	1.12	1.23	1.13
3*c*	N_{mil}	516	423	461	690	1335	2319	4172	540	696	543
	P_{mil}	1.22	1.00	1.09	1.63	3.16	5.48	9.87	1.28	1.65	1.28
	N_{col}	102561	83150	82067	93159	118517	145181	194409	85648	93408	86068
	P_{col}	1.25	1.01	1.00	1.14	1.44	1.77	2.37	1.04	1.14	1.05
4*a*	N_{mil}	17046	16510	17514	14361	16523	26995	30000	14303	14371	14234
	P_{mil}	1.20	1.16	1.23	1.01	1.16	1.90	2.11	1.00	1.01	1.00
	N_{col}	3.7e+6	3.4e+6	3.3e+6	2.2e+6	2.2e+6	3.6e+6	3.7e+6	2.2e+6	2.2e+6	2.2e+6
	P_{col}	1.71	1.56	1.55	1.01	1.03	1.65	1.73	1.00	1.01	1.00
4*b*	N_{mil}	5188	3797	2975	2514	4433	8150	14509	2459	2488	2738
	P_{mil}	2.11	1.54	1.21	1.02	1.80	3.31	5.90	1.00	1.01	1.11
	N_{col}	1.5e+6	1.0e+6	0.7e+6	0.5e+6	0.7e+6	1.1e+6	1.7e+6	0.5e+6	0.5e+6	0.6e+6
	P_{col}	3.21	2.23	1.55	1.01	1.52	2.38	3.69	1.00	1.01	1.27
	$\overline{P}_{\text{mil}}$	3.54	2.20	1.52	1.35	2.00	3.30	5.25	1.22	1.34	1.25

6.2 The Behaviors of Adaptive Sampling Strategies

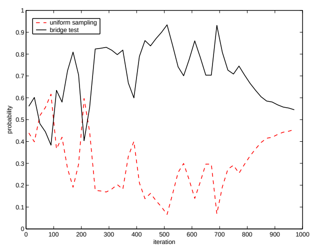

Fig. 5. The change of probabilities for combining samplers.

To gain some intuition on the behaviors of adaptive sampling strategies, let us look at how the probabilities for combining the component samplers change over time (Figure 5). The plot was obtained for sampler AS3 in an environment shown in Figure 2*c*. Figure 5 shows that in the first 100 iterations, the uniform sampler and the bridge test have roughly the same probabilities of being chosen. The reason is that since the free space contains both large collision-free regions and narrow passages, the two samplers perform equally well. In the next 600 iterations, the uniform sampler is no longer effective, as most of the free space has already been covered, but the bridge test still obtains useful milestones from the narrow passages. As a result, the bridge test has much higher weight. Finally when the roadmap contains enough milstones, neither samplers can obtain additional useful milestones easily. So their weights become more similar again.

Table 2. The performance of various hybrid sampling strategies with a new reward function.

Env.		FHS	FHS	FHS	FHS	FHS	FHS	FHS	AS3	AS2	AS1
(Fig.)		1/16	1/8	1/4	1/2	3/4	7/8	15/16			
3*a*	N_{col}	73156	42341	23800	15624	17066	17791	16613	16137	15543	15505
	P_{col}	4.72	2.73	1.53	1.01	1.10	1.15	1.07	1.04	1.00	1.00
	N_{col}	98774	54648	33494	18014	11941	10214	9378	13632	18100	13347
	P_{col}	10.53	5.83	3.57	1.92	1.27	1.09	1.00	1.45	1.93	1.42

7 Discussion

Our current adaptive sampling strategies use a reward function that minimizes the number of milestones in the roadmap, because a small roadmap speeds up the query processing and often leads to fast pre-computation as well. However, if pre-computation time is the main objective, then we should explicitly incorporate into the reward function the cost of obtaining a milestone and adding it to the roadmap and minimize the ratio of reward over cost. We did some initial testing with this new reward function on the two environments in which the original reward function has unsatisfactory pre-computation time. The results, shown in Table 2, indicate that the new reward function indeed improves pre-computation time, if we compare the results with the corresponding ones for the last two environments in block 3*a* of Table 1. We are currently exploring efficient ways to implement this new reward function as well as other interesting reward functions.

8 Conclusion and Future Work

In this paper, we propose a systematic approach for adaptively combining multiple sampling strategies for PRM planning. Using this approach, we describe three adaptive hybrid sampling strategies. They were tested on robots with two to eight dofs in planar workspaces. In these preliminary tests, the adaptive planners showed consistently good performance, compared with fixed-weight hybrid-sampling planners. In particular, AS3, the simplest adaptive planner, performed surprisingly well.

For future work, we plan to apply adaptive hybrid sampling to 3-D workspaces to examine it effectiveness further. More importantly, we plan to use a large number of component samplers and investigate which one of our adaptive strategies, or any new ones, work well. We are also interested in applying this approach to single-query motion planning problems.

Acknowledgements We thank Lee Wee Sun for pointing us to the work in computational learning theory and for many helpful discussions.

References

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

2. P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Computing*, 32(1):48–77, 2002.
3. A. Blum. On-line algorithms in machine learning. Technical Report CMU-CS-197-163, Carnegie Mellon University, 1997.
4. R. Bohlin and L. Kavraki. Path planning using lazy PRM. In *Proc. IEEE Int. Conf. on Robotics & Automation*, pages 521–528, 2000.
5. 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*, pages 1018–1023, 1999.
6. M. Branicky, S. LaValle, K. Olson, and L. Yang. Quasi-randomized path planning. In *Proc. IEEE Int. Conf. on Robotics & Automation*, pages 1481–1487, 2001.
7. L. Dale and N. Amato. Probabilistic roadmaps—putting it all together. In *Proc. IEEE Int. Conf. on Robotics & Automation*, pages 1940–1947, 2001.
8. 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*, pages 55–60, 2001.
9. L. Guibas, C. Holleman, and L. E. Kavraki. A probabilistic roadmap planner for flexible objects with a workspace medial-axis based sampling approach. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, pages 254–260, 1999.
10. 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*, pages 1408–1413, 2000.
11. 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*, pages 4420–4426, 2003.
12. D. Hsu, J. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *Int. J. Computational Geometry & Applications*, 9(4 & 5):495–512, 1999.
13. L. Kavraki, P. Švestka, J. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration space. *IEEE Trans. on Robotics & Automation*, 12(4):566–580, 1996.
14. S. LaValle and J. Kuffner. Randomized kinodynamic planning. *Int. J. Robotics Research*, 20(5):278–400, 2001.
15. 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*, pages 4439–4444, 2003.
16. N. Littlestone and M. Warmuth. The weighted majority algorithm. *Information & Computation*, 108(2):212–261, 1994.
17. C. Nissoux, T. Siméon, and J.-P. Laumond. Visibility based probabilistic roadmaps. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, pages 1316–1321, 1999.
18. G. Sanchez and J. Latombe. On delaying collision checking in PRM planning—application to multi-robot coordination. *Int. J. Robotics Research*, 21(1):5–26, 2002.
19. R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.
20. S. Wilmarth, N. Amato, and P. Stiller. MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space. In *Proc. IEEE Int. Conf. on Robotics & Automation*, pages 1024–1031, 1999.