Incremental Map Generation (IMG)

Dawen Xie, Marco A. Morales A., Roger Pearce, Shawna Thomas, Jyh-Ming Lien, and Nancy M. Amato

Parasol Lab, Department of Computer Science,
Texas A&M University, College Station, TX USA
{dawenx, marcom, rap2317, sthomas, neilien, amato}@cs.tamu.edu

Abstract: Probabilistic roadmap methods (PRMs) have been highly successful in solving many high degree of freedom motion planning problems arising in diverse application domains such as traditional robotics, computer-aided design, and computational biology and chemistry. One important practical issue with PRMs is that they do not provide an automated mechanism to determine how large a roadmap is needed for a given problem. Instead, users typically determine this by trial and error and as a consequence often construct larger roadmaps than are needed. In this paper, we propose a new PRM-based framework called Incremental Map Generation (IMG) to address this problem. Our strategy is to break the map generation into several processes, each of which generates samples and connections, and to continue adding the next increment of samples and connections to the evolving roadmap until it stops improving. In particular, the process continues until a set of evaluation criteria determine that the planning strategy is no longer effective at improving the roadmap. We propose some general evaluation criteria and show how to apply them to construct different types of roadmaps, e.g., roadmaps that coarsely or more finely map the space. In addition, we show how IMG can be integrated with previously proposed adaptive strategies for selecting sampling methods. We provide results illustrating the power of IMG.

1 Introduction

Automatic motion planning has applications in many areas such as robotics, computer animation, computer-aided design (CAD), virtual prototyping, and computational biology and chemistry. Although many deterministic motion planning methods have been proposed, most are not used in practice because they are computationally infeasible except for some restricted cases, e.g., when the robot has few degrees of freedom (dof) [16]. Indeed, there is strong evidence that any complete planner (one that is guaranteed to find a solution or determine that none exists) requires time exponential in the robot's dof [23].

For this reason, attention has focused on randomized approaches that sample and connect points in the robot's configuration space (C-space). Such

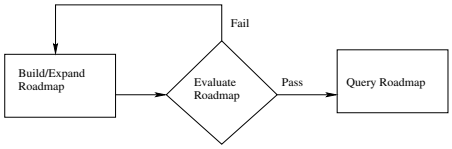

Fig. 1. Flow diagram for Incremental Map Generation (IMG).

methods include graph-based methods such as the *probabilistic roadmap methods* (PRMs) [15] (along with their various extensions and variants [1,4,5,11,27]) and tree-based methods such as Ariadne's Clew algorithm [18], RRT [17], and Hsu's expansive planner [12]. These methods have been highly successful in solving challenging problems with many dof that were previously unsolvable and thus have become the method of choice for a wide range of applications.

One important practical issue not addressed by the PRM framework is that it does not provide an automated mechanism to determine when to stop. Ideally, planning should be terminated when the planner is no longer adding useful information to the roadmap. In practice, users select a roadmap size they believe is appropriate, usually by trial and error. This often results in larger maps than needed or in the construction of several maps before obtaining one that meets the user's needs. While there are a number of reasons for this disconnect between the ideal and practice, perhaps the most important has been the lack of effective techniques for measuring roadmap improvement.

In this paper, we propose a PRM-based framework called Incremental Map Generation (IMG) that addresses this issue. In particular, we advocate a strategy that measures the improvement achieved over time in an evolving roadmap to automatically determine when to stop (or perhaps change) the planner. This is implemented by iteratively building the roadmap until it satisfies a set of evaluation criteria (see Figure 1). The main difference from the traditional two-phase PRM method [15] is that we partition the roadmap construction into several iterations (expansion steps), each of which adds samples and connections to the evolving roadmap, and we add a new phase called "roadmap evaluation" which tests if the roadmap satisfies some evaluation criteria (stopping condition). If the roadmap passes the stopping condition, then roadmap construction finishes. Otherwise, another iteration is performed to expand the roadmap by adding additional samples and connections. The framework can accept a broad range of stopping criteria, which can be customized for particular applications or user preferences. For example, the criteria can be as simple as satisfying a specified set of queries, or more complicated such as monitoring graph topology.

IMG has several important features, including:

- *Automatic determination of roadmap size.* The most important feature of IMG is that it provides a mechanism to incrementally construct roadmaps and to automatically determine when construction should be halted.

- *Evaluation criteria.* A key requirement for IMG is effective evaluation criteria that can be efficiently tested during roadmap construction. A contribution of this work is to propose evaluation criteria for measuring roadmap quality (e.g., coverage and connectivity) that do not require prior knowledge about the solution (as do, e.g., test queries) and that do not rely on C-space discretization (so can be efficiently applied to high dof problems).
- *Compatibility with existing sampling-based planners.* IMG is *not* a new sampling method; instead, it is a general strategy that can be used with any sampling-based planner and, moreover, provides a natural mechanism for adaptive planning. For example, each IMG iteration can use any of the existing adaptive strategies that utilize multiple planners (e.g., [13,19]) or different strategies could be chosen for different IMG iterations.

2 Related work

The general PRM methodology [15] consists of a preprocessing phase and a query phase. Preprocessing, which is done once for a given environment, first samples points 'randomly' from the robot's C-space, retaining those that satisfy certain feasibility requirements. Then, these points are connected to form a graph, or roadmap, containing representative paths in the free C-space. The query phase then connects any given start and goal to the same connected component of the roadmap, and if successful, returns a path connecting them.

The probability of failing to find a path in a probabilistic roadmap, when one exists in C-free, decreases exponentially as the number of samples in the roadmap increases [14]. However, it is difficult to decide beforehand the roadmap size required in practice.

The *coverage* and *connectivity* of an ideal PRM roadmap should match that of its underlying C-space. In [9], coverage and maximal connectivity achieved by different sampling methods was compared to that of the C-space being modeled. Coverage indicates how each query can be connected to the roadmap. If there exists a path in the free C-space between two query configurations, maximal connectivity ensures that a path between them can be found in the roadmap. The authors evaluate the time needed to adequately cover and connect the free C-space for various techniques. This work relies on a discretization of C-space and so cannot be applied to high dof problems.

Since there is no principled mechanism to determine when to stop roadmap construction, a commonly used evaluation criterion is to predefine a set of relevant queries in each environment and continue building the roadmap until the query configurations can be connected to the same connected component. This is helpful in environments where the user knows beforehand such a representative query. However, in many situations defining such a query can be problematic, e.g., in cluttered environments or in higher dof problems such as the protein folding applications. The stopping criteria we propose can be applied when it is hard or even impossible to define a representative query

for a given problem. In the case when such a query is easy to define or when solving a particular query is the user's objective, then IMG can easily use it as a stopping criterion, i.e., IMG also supports the traditional query-based criterion.

A set of metrics are proposed in [20] to estimate how each new sample improves, or not, the representation of C-space achieved by the planner. With these metrics, the authors identify three phases common to all sampling-based planners: quick learning (a coarse roadmap is constructed), model enhancement (the roadmap is refined), and learning decay (most new samples do not provide additional information). They also demonstrate that the traditional scheme of testing a set of witness queries, which is commonly used in practice as a stopping criteria, can be misleading.

Adaptive hybrid PRM sampling [13] proposes using a mixture of samplers. They adapt the mixture of strategies based on each strategy's past success. In this work, we incorporate hybrid PRM sampling and apply our IMG framework to hybrid PRM to decide when to stop building the roadmap.

3 Incremental Map Generation (IMG)

We propose a new PRM-based framework called Incremental Map Generation (IMG) in which we iteratively build a roadmap until it satisfies a set of evaluation criteria (see Figure 1). Most importantly, this framework provides a systematic way to automatically decide when to stop roadmap construction. Algorithm 3.1 describes IMG. This framework is simple and general. It can be customized for a particular application domain or problem by simply varying the node generation and connection strategies used and the evaluation criteria. In the following sections we discuss two main aspects of our framework: incremental roadmap construction and roadmap evaluation.

Algorithm 3.1 Incremental Map Generation.

Input. An existing roadmap R, a roadmap evaluator E, the size of a node set, n.
Output. A roadmap R that meets the criteria indicated by E.

1: **repeat**
2: *Initialization.* Set parameters for this iteration.
3: *Sampling.* Generate the new node set (n nodes) and add them to roadmap R.
4: *Connection.* Perform connection.
5: **until** R meets criteria in E

3.1 Incremental Roadmap Construction

To build the roadmap incrementally, we first divide roadmap construction into "sets" of size n; the size, or target number of nodes for each set, is specified by the user. Then, for each iteration, IMG performs the following steps.

Initialization. In line 2, Algorithm 3.1, in order to ensure the independence of each set, we seed the random number generator. The seed s is a polynomial function of the *base seed* of the program (e.g., the time execution starts), the type of node generation method used, and the number of sets completed by that node generation method so far. Calculating the seed in a deterministic way based on a (possibly random) base seed supports reproducibility given the same base seed.

Sampling. In line 3, Algorithm 3.1, the sampling strategy selected for that iteration is applied. Recall that IMG is *not* a new sampling method, but rather is a general strategy that can be applied to any sampling-based planner.

In addition, IMG provides a natural mechanism for adaptive planning. For example, each iteration of IMG could select a different sampling strategy or it might use one of the recently proposed adaptive strategies that utilize multiple planners (e.g., [13, 19]). To illustrate this feature of IMG, we incorporated hybrid PRM [13] in our current implementation. In hybrid PRM [13], the performance of component samplers is evaluated and the methods with good performance are chosen to run more frequently. We incorporated hybrid PRM in two different ways: (1) it is simply used as described in [13] as the sampling method in IMG, and (2) in each IMG iteration, an initial phase uses the hybrid PRM strategy to select a planner to use for the remainder of that iteration.

Connection. In line 4, Algorithm 3.1, the connection strategy chosen by the user is applied to connect the new set of nodes to the existing roadmap. In the results presented here, we use a variant of the commonly used K-closest connection strategy. K-closest attempts to connect each node to its k "nearby" neighbors, but it does not distinguish successful attempts from failed attempts. Nevertheless, identifying successes and failures in connection attempts provides some information about the complexity of the local area. When a node can be connected to most of its neighbors, it indicates that this node is in an easy to connect area and we probably do not need to try many connection attempts; on the other hand, if a node fails to be connected to most of its neighbors, it indicates that this node is in a difficult local area and it could be useful to try to connect it to more neighbors. In order to adjust the connection effort based on a node's local environment, we use a modified version of K-Closest connection method called *L-Success-M-Failure*. In L-Success-M-Failure, the local planner attempts to connect each node to its $l + m$ "nearby" neighbors, stopping as soon as it has achieved l successful attempts or m failed attempts.

3.2 Roadmap Evaluation

The other key component enabling automatic determination of roadmap size is the stopping or evaluation criteria. In this paper, we propose two classes of evaluation methods: *roadmap progress evaluation* and *application-specific evaluation*. The following sections give examples of evaluators for both classes.

Roadmap Progress Evaluation

Our roadmap progress evaluators are based on metrics for evaluating roadmap *coverage* and *connectivity*, which have been noted as important properties by many researchers working with sampling-based planners (see, e.g., [9]). Here, we are interested in monitoring the contribution of new samples to the coverage and connectivity modeled by the roadmap. Classification of new samples provides a mechanism to perform this evaluation as shown in [20]. In [20], every node is classified as it is inserted into the roadmap (see Figure 2). A node is classified as *cc-create* if it cannot be connected to any existing roadmap component. A node is classified as *cc-merge* if it connects to more than one connected component (CC) in the roadmap. A node is classified as *cc-expand* if it connects to exactly one component in the roadmap and satisfies an expanding criterion. A node is classified as *cc-oversample* if it does not fall in any of the previous categories. In previous work [13, 22], cc-expand and cc-oversample nodes were not always distinguished, in some cases because it was considered too expensive to classify a node as cc-expand.

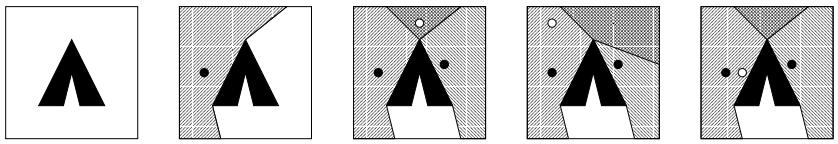

Fig. 2. (a) A 2D C-space. Classification of samples as (b) cc-create, (c) cc-merge, (d) cc-expand, and (e) cc-oversample.

In this work, we use the diameter of the connected components as a measurement of component expansion. The *diameter* of a CC is the length of the longest shortest-path in the CC. The diameter is an interesting metric because it may increase by adding cc-create, cc-merge or cc-expand nodes but not from adding cc-oversample nodes. Also, the diameter of a graph can be approximated and is independent of the distance metric [24]. We let *max-diameter* be the maximum diameter of all the CCs and *sum-diameter* be the sum of the diameters of all CCs. Note that max-diameter is an approximation of the coverage of the largest connected component. Similarly, sum-diameter is an approximation of the coverage and connectivity of all the components in the roadmap. Then we use the rates of change of max-diameter and sum-diameter to approximate the planner's effectiveness in mapping C-space.

In this evaluation method, we stop building a roadmap when the rate of change of max-diameter and sum-diameter over a certain period of time, e.g., k sets of nodes, is smaller than a user-defined threshold, τ, which is used to define the desired variability in coverage and connectivity (as indicated by the

components' diameters). We compute the max-diameter among all CCs and the sum-diameter of all the CCs at the end of each node set.

The percentage of change of the max-diameter ($PCMAX_i$) in the i^{th} set over its k previous sets is computed as:

$$PCMAX_i = \sum_{j=0}^{k-1} \frac{|MD_{i-j} - MD_{i-j-1}|}{MD_{i-j-1}},$$

where MD_{i-j} is the max-diameter in the $(i-j)^{th}$ node set. We define the percentage change of the sum-diameter ($PCSUM_i$) over all the components in a similar way.

Application-Specific Evaluation

The IMG framework can accept a broad range of stopping or evaluation criteria customized for particular applications or user preferences. In this section, we give two examples of application-specific evaluation methods.

Query Evaluation. This evaluator simply determines whether a roadmap can solve a set of user specified queries. For each query, it attempts to connect the start and goal to the roadmap and returns successful if they are connected to the same connected component. The evaluator returns success when all queries are solved. This type of evaluator is useful when the user wants to solve a particular set of test problems or for a single query application.

Max-flow Evaluation. Some applications require many paths between two configurations. For example, motion planning has been recently applied to study problems in computational biology such as protein folding and transitions [3, 25]. To study how a protein changes between two configurations, we can examine the probable paths between them in the roadmap. We can define this as a maximum flow problem on a network. If a roadmap edge weight, $w(e)$, reflects the likelihood that the protein will move from one configuration to the next, then we can define edge capacity $c(e)$ as $1/w(e)$. The evaluator returns success if the max-flow between the two configurations is above some user specified threshold f.

4 Experiments

IMG is *not* a new sampling method, instead it is a general strategy that can be applied to any sampling-based planner. We investigate how IMG automatically builds roadmaps with an appropriate number of samples using different evaluation criteria. Our experiments use the following sampling methods:

- *Uniform random sampling*: samples are created by picking random values for each dof.

- *Gaussian-biased sampling* [4]: sets of two samples are created, one uniformly at random and the other a distance d away, where d has a Gaussian distribution. A collision-free sample is added to the roadmap when one is collision-free and the other is not.
- *Bridge-test sampling* [11]: similar to Gaussian sampling, it takes two random samples a distance d apart, where d has a Gaussian distribution, until both samples are in collision and their midpoint is not. The collision-free sample is added to the roadmap.
- *Obstacle-based sampling* (OBPRM) [1]: samples are generated near C-obstacle surfaces by first generating a random colliding (resp., collision-free) sample and searching along a random direction until the sample becomes collision-free (resp., in collision).

We implemented all planners with the Parasol Lab motion planning library developed at Texas A&M University and performed collision detection with RAPID [10]. For each problem, we built two types of roadmaps: a tree and a graph. We use the L-Success-M-Failure connection strategy introduced in Section 3. In particular, we apply a 10-Success-20-Failure connection strategy for building trees and 5-Success-20-Failure for building graphs. For rigid-body motion planning, we use two local planners: straight-line and rotate at 0.5 [2], which translates from the start to the midpoint, rotates to the orientation of the goal configuration and then translates to the goal configuration. For articulated linkage motion planning, we only use the straight-line local planner. All results were run on 700MHz Intel PIII Xeon processors.

In the following sections, we discuss the performance of IMG's roadmap progress evaluator, the overhead of the IMG framework, how IMG and hybrid PRM may be combined, and how IMG can be tailored to specific applications such as protein folding.

4.1 Automatically Stopping Roadmap Construction

Here we investigate the performance of the roadmap progress evaluator (see Section 3.2) in the four different environments shown in Figure 3. For these experiments, the node set size is 50 samples. After each set, we compute $PCMAX_i$ and $PCSUM_i$. Roadmap construction stops when both $PCMAX_i$ and $PCSUM_i$ are below a threshold τ. τ represents the desired roadmap improvement over a period of time, i.e., k sample sets. Note that in the beginning of roadmap construction (during the quick learning stage), there will be large changes in $PCMAX$ and $PCSUM$. These changes will drop when the enhancement stage begins.

We studied the impact of τ and k on IMG's performance for each sampling method in each environment by varying k with constant τ and alternatively by varying τ with constant k. Due to space limitations, we only show a subset of these results. A complete set of results can be found on our website.

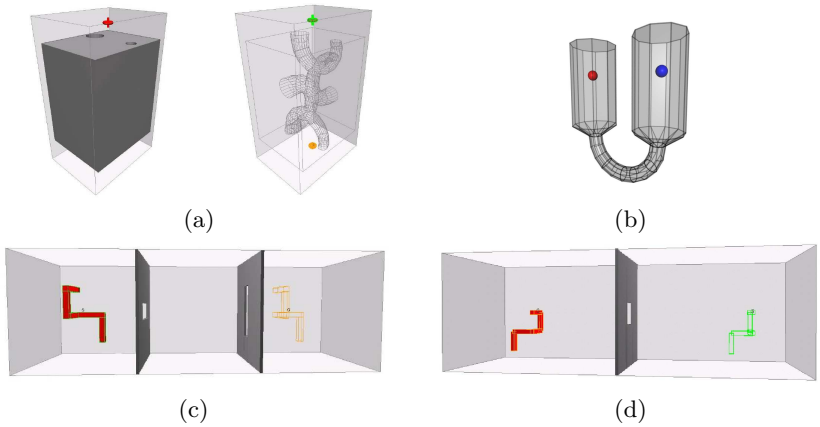

Fig. 3. Problems studied. (a) Maze environment (solid/wire frame): rigid body robot must navigate the maze. (b) U shape environment: rigid body robot must navigate from one chamber to the other. (c) Hook environment: rigid body robot must rotate to move from one side of the walls to the other. (d) Hook manipulator environment: articulated linkage (10 dof) must move from one end to the other.

A Case Study: Varying k with Constant τ

Figure 4 shows IMG's performance at building both trees and graphs for each planner in the hook environment. Here we vary k (the number of sample sets over which the percentage change in diameter is computed) while keeping τ constant at 0.0125. In each plot, the upper two curves show the sum-diameter and max-diameter as a function of roadmap size for a tree, and the lower two curves for a graph. The circles indicate where IMG would stop roadmap construction for various k. For example, the circle labeled $k_1, 1250$ in Figure 4(a) shows that with $k = k_1 = 5$, IMG would stop construction of the tree after 1250 samples. Similarly, the circle labeled $k_1, 2150$ indicates IMG would stop construction of the *graph* after 2150 samples with the same k value. All plots use the same random seed.

From the evolution of max-diameter and sum-diameter, it is clear that the roadmap grows rapidly in the beginning and then experiences a long period of refinement until both stabilize. As expected, the diameter in the tree roadmap is larger than the diameter in the graph roadmap. This corresponds to the graph roadmap having shorter and smoother paths. An interesting observation from the graph roadmap is that the "path refinement" stage is clearly shown as the diameters drop.

Overall, we see that for a fixed τ, increasing k causes the planner to stop later because larger k values allow IMG to capture changes over longer periods. This trend appears in all experiments we ran. This means that we can decide how long we want to refine the roadmap by what value we choose for k. It is also clear that for a given k value, different planners stop at different points.

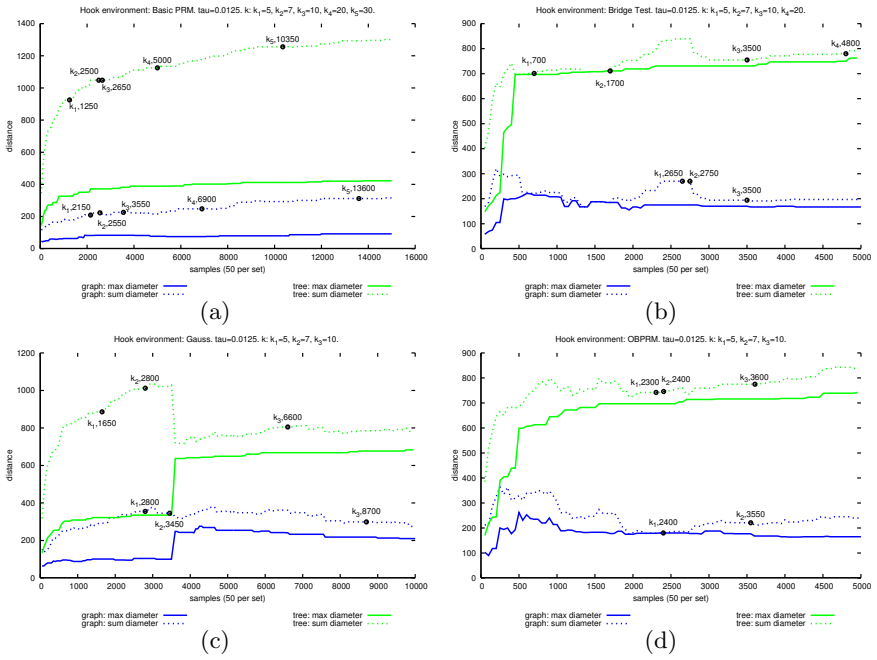

Fig. 4. Performance of IMG for the hook environment for various planners and k values with $\tau = 0.0125$. (a) BasicPRM; unable to solve the witness query with 15000 samples. (b) Bridge test; witness query solved at 550 samples. (c) Gauss; witness query solved at 3600 samples. (d) OBPRM; witness query solved at 500 samples.

In particular, BasicPRM (Figure 4(a)) stops the earliest. The intuition behind this is that BasicPRM is the slowest to progress in terms of samples, and thus needs larger values of k to capture similar changes.

Finally, we defined a witness query from the first chamber to the last chamber. We use this query in the query evaluation method as described in Section 3.2. BasicPRM is unable to solve the query after 15000 samples, while Bridge test solved it after 550 samples, Gauss after 3600 samples, and OBPRM after 500 samples[1]. It is clear from Figure 4 that the max-diameter and sum-diameter still experience large changes after solving the witness query. This confirms the observation in [20] that solving queries is not enough by itself to evaluate whether the planner is still making progress in mapping the space.

A Case Study: Varying τ with Constant k

Here we study the effect of τ (a threshold for desired variability in coverage and connectivity) while fixing k at 10 for OBPRM [1] in different environments.

[1] The tree and graph roadmaps solve the query at the same point since we used the same random seed.

Figure 5 shows results for both trees and graphs using the same random seed. As before, the circles indicate where IMG would stop roadmap construction for the various τ.

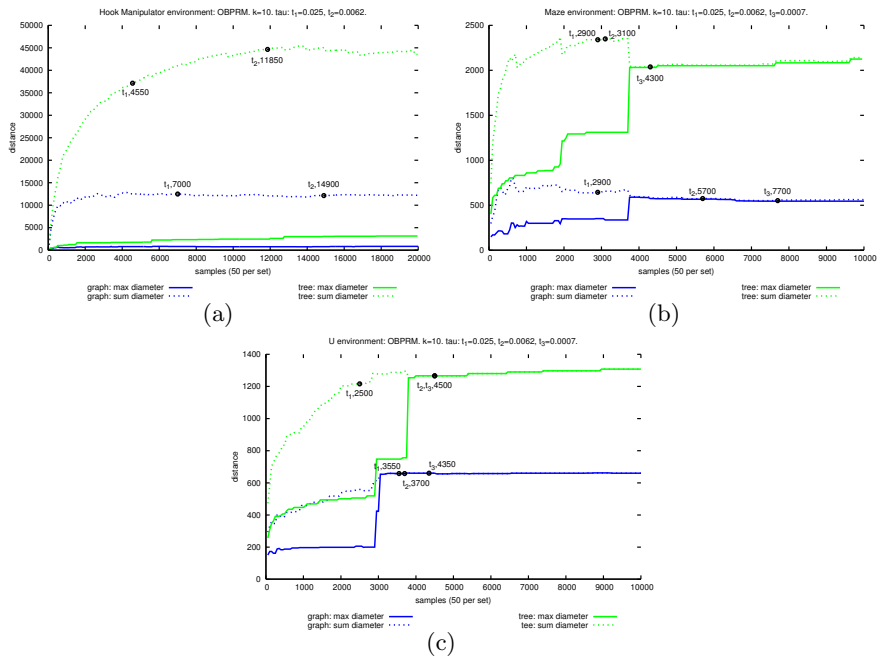

Fig. 5. Performance of IMG with OBPRM for several environments and τ values when $k = 10$. (a) Hook manipulator environment; unable to solve witness query in 20000 samples. (b) Maze environment; witness query solved at 3750 samples. (c) U environment; witness query solved at 2850 samples.

Overall, decreasing τ requires the planner to run longer because smaller τ values signify a smaller tolerance for diameter variability. Thus, a smaller τ means the planner has to run longer before the learning stabilizes enough to cause the diameter changes to fall below the threshold.

We set witness queries as described in Figure 3. OBPRM is unable to solve the query for the hook manipulator after 20000 samples, while it solved the query for the maze environment after 3750 samples and the U environment after 2850 samples. For the hook manipulator (Figure 5(a)), we observe that $\tau = 0.025$ roughly marks the end of the "quick learning" stage as it transitions into "model enhancement." Note, the planner remains in "model enhancement" for the entire duration. This is reflected by the fact that $\tau = 0.0007$ was never satisfied and the witness query was never solved. In Figures 5(b) and 5(c), "learning decay" is clearly marked with $\tau = 0.0007$.

Overhead

The overhead incurred by the calculation of roadmap diameter as an evaluation of roadmap progress is affordable. We show in Table 1 the percentage of total running time spent in the diameter computation for the hook environment for the case when the roadmap can contain cycles (a graph). The diameter computation is performed after every 50 samples and for the tree roadmap case is exactly computed by a Dijkstra search. In the cyclic graph case, the diameter is approximated using two Dijkstra searches, with the second search starting from the furthest node found during the first search. While more accurate approximations of cyclic graph diameters exist [6], this was sufficient for our experiments.

Table 1. Diameter computation as a percentage of total running time in the Hook environment. Diameter computation was performed after every 50 samples. For all methods, the overhead for IMG is small, even for large numbers of samples.

Sampling Method	**Number of Samples**					
	100	500	1000	2000	4000	8000
BasicPRM	0.20	1.04	1.86	3.16	4.75	6.50
OBPRM	0.00	0.44	0.93	1.70	2.84	4.41
Gauss	0.09	0.48	0.95	1.78	3.13	5.39
Bridge test	0.02	0.09	0.19	0.45	0.99	1.93
Hybrid PRM	0.08	0.34	0.71	1.47	2.70	4.64

Combining IMG and Hybrid PRM

The IMG framework can be used with any sampling strategy. In this section, we incorporated hybrid PRM [13] in two different ways. First, we simply used hybrid PRM as the sampling strategy in IMG. Second, we partitioned each IMG iteration into two parts: a "learning window" and a "sampling window," with the learning window at 20% of the total set size. During the learning window, we use hybrid PRM to learn the appropriate probabilities of using each sampler, starting with a uniform distribution. We then fix this distribution during the sampling window. We experimented with several different ways of learning during the learning window and all variants displayed comparable behavior. In the results shown here, we use a fixed uniform probability distribution during the learning window, but learn the probability distribution for the sampling window just as with hybrid PRM sampling. Because the learning window is relatively small, this allows the learner to observe all the samplers. For all experiments shown here, we use the cost-based version of hybrid PRM described in [13] and five component samplers: BasicPRM, two versions of

the Bridge test, and two versions of Gauss. The reward mechanism of the original hybrid PRM only rewards a planner when it generates *cc-create* and *cc-merge* samples. However, a sample that expands a roadmap is also important. Therefore, when a planner generates a *cc-expand* sample we give it a reward equal to the complement of the percentage of successful connections from that sample to the roadmap.

In Figure 6, we show hybrid PRM in the IMG framework. For clarity, we only show BasicPRM, and the version of Bridge test and Gauss that performed best. Figure 6(a) shows where IMG would stop roadmap construction with pure hybrid PRM sampling for various values of k and τ. This shows similar trends when varying k and τ as seen previously. Figure 6(b) shows the number of samples created by each sampler vs. the total number of samples. Figure 6(c) shows the probability of being selected along with the percentage of cc-oversample nodes for each sampler during roadmap construction. Our results here confirm the results found in [13]: BasicPRM is selected early on because it is a relatively inexpensive sampler but dies out quickly as other, more powerful and expensive samplers are selected. In the end, after the witness query is solved, hybrid PRM vacillates between a version of Gauss and a version of Bridge test.

Figure 6(d) and 6(e) show similar plots as 6(b) and 6(c), respectively, for IMG with a hybrid learning window. Unlike the previous plots (b and c), this version does not select a dominant sampler towards the end of roadmap construction, after the witness query is solved. We believe in fact that this is a more accurate evaluation because at this stage all samplers are equally "bad," i.e., none are able to generate useful samples and should not necessarily be distinguished. In particular, note that more than 80% of the nodes created by all samplers are cc-oversample nodes in the later stages of roadmap construction.

4.2 Application-Specific Stopping Criteria

As discussed in Section 3.2, the IMG framework can accept a broad range of stopping criteria that can be customized for particular applications or user preferences. We can apply our framework to study computational biology problems such as protein folding and protein structure transitions. Here, we incrementally build a roadmap until the maximum flow between the two configurations of interest is above a threshold. We applied this approach to study calmodulin, a signaling protein that binds to Ca^{2+} to regulate several processes in the cell [7, 8, 26]. When calmodulin binds to Ca^{2+}, it undergoes a large-scale rearrangement [21] shown in Figure 7.

Using IMG, we incrementally built a roadmap until it adequately described the C-space around both target configurations as well as the transition between. We were able to build the smallest roadmap possible, given the increment size, in 2 weeks of computation time. The traditional method of building a small roadmap, evaluating it, and rebuilding a larger one until it adequately describes the C-space would take at least 4 weeks of computation time.

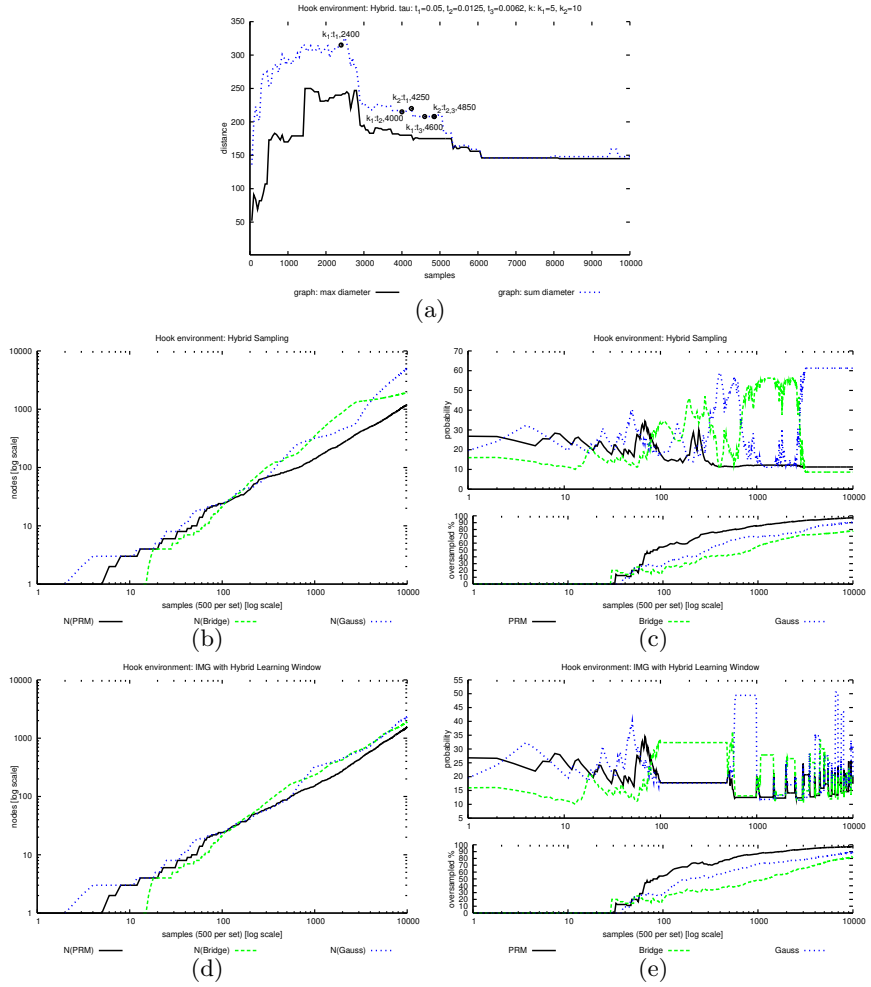

Fig. 6. Applications of Hybrid PRM with IMG. Hybrid PRM using IMG: (a) stopping criteria, (b) number of samples per sampler, and (c) sampler probability and oversample %. Hybrid PRM Learning Window: (d) number of samples per sampler and (e) sampler probability and oversample %. The witness query solved after 1440 samples.

5 Conclusion

In this paper, we proposed a framework to automatically determine how many samples a planner needs to construct for a given motion planning problem. This framework can accept a broad range of evaluation criteria which can be customized for particular applications. We provide easy to define parameters that allow users to stop roadmap construction by satisfying criteria based on

(a) (b)

Fig. 7. Rearrangement of Calmodulin: (a) calcium-free state to (b) bound state.

the quality of the roadmap. This has many potential applications that we plan to study. There are also several other areas that we would like to investigate further. First, we would like to expand our list of node generation methods to include other types of random sampling and grid-based techniques. Second, in the computational biology application, we want to replace the diameter of the CC with other energetically meaningful metrics, e.g., potential energy, as a measurement of a component's expansion to guide the planner to construct a more energetically feasible roadmap.

References

1. N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo. OBPRM: An obstacle-based PRM for 3D workspaces. In *Robotics: The Algorithmic Perspective*, pages 155–168, Natick, MA, 1998. A.K. Peters. Proc. Third Workshop on Algorithmic Foundations of Robotics (WAFR), Houston, TX, 1998.
2. N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo. Choosing good distance metrics and local planners for probabilistic roadmap methods. *IEEE Trans. Robot. Automat.*, 16(4):442–447, August 2000.
3. M. Apaydin, A. Singh, D. Brutlag, and J.-C. Latombe. Capturing molecular energy landscapes with probabilistic conformational roadmaps. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 932–939, 2001.
4. V. Boor, M. H. Overmars, and A. F. van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 2, pages 1018–1023, 1999.
5. B. Burns and O. Brock. Sampling-based motion planning using predictive models. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2005.
6. D. G. Corneil, F. F. Dragan, and E. Köhler. On the power of bfs to determine a graph's diameter. *Networks*, 42(4):209–222, 2003.
7. A. Crivici and M. Ikura. Molecular and structural basis of target recognition by calmodulin. *Annu. Rev. Biophys. Biomol. Struct.*, 24:85–116, 1995.
8. J. Evenas, A. Malmendal, and S. Forsen. Calcium. *Curr. Op. Chem. Biol.*, 2(2):293–302, 1998.
9. R. Geraerts and M. H. Overmars. Reachablility analysis of sampling based planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 406–412, 2005.

10. S. Gottschalk, M. C. Lin, and D. Manocha. OBB-tree: A hierarchical structure for rapid interference detection. *Comput. Graph.*, 30:171–180, 1996. Proc. SIGGRAPH '96.
11. D. Hsu, T. Jiang, J. Reif, and Z. Sun. Bridge test for sampling narrow passages with proabilistic roadmap planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 4420–4426, 2003.
12. D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2719–2726, 1997.
13. D. Hsu, G. Sánchez-Ante, and Z. Sun. Hybrid PRM sampling with a cost-sensitive adaptive strategy. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3885–3891, 2005.
14. L. E. Kavraki, J.-C. Latombe, R. Motwani, and P. Raghavan. Randomized query processing in robot path planning. In *Proc. ACM Symp. Theory of Computing (STOC)*, pages 353–362, May 1995.
15. L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.*, 12(4):566–580, August 1996.
16. J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.
17. S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 473–479, 1999.
18. E. Mazer, J. M. Ahuactzin, and P. Bessiere. The Ariadne's clew algorithm. In *Journal of Artificial Robotics Research (JAIR)*, volume 9, pages 295–316, 1998.
19. M. Morales, L. Tapia, R. Pearce, S. Rodriguez, and N. M. Amato. A machine learning approach for feature-sensitive motion planning. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 316–376, Utrecht/Zeist, The Netherlands, July 2004.
20. M. A. Morales A., R. Pearce, and N. M. Amato. Metrics for comparing C-Space roadmaps. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2006.
21. M. R. Nelson and W. J. Chazin. An interaction-based analysis of calcium-induced conformational changes in ca2+ sensor proteins. *Protein Sci.*, 7:270–282, 1998.
22. C. Nissoux, T. Simeon, and J.-P. Laumond. Visibility based probabilistic roadmaps. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 1316–1321, 1999.
23. J. H. Reif. Complexity of the mover's problem and generalizations. In *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, pages 421–427, San Juan, Puerto Rico, October 1979.
24. R. Seidel. On the all-pairs-shortest-path problem. In *Proc. 24th Annu. ACM Sympos. Theory Comput.*, pages 745–749, 1992.
25. G. Song and N. M. Amato. Using motion planning to study protein folding pathways. In *Proc. Int. Conf. Comput. Molecular Biology (RECOMB)*, pages 287–296, 2001.
26. H. Vogel. Calmodulin: a versatile calcium mediator protein. *Biochem. Cell Biol.*, 72(9-10):357–376, 1994.
27. S. A. Wilmarth, N. M. Amato, and P. F. Stiller. MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 2, pages 1024–1031, 1999.