Incremental Map Generation (IMG)

Dawen Xie, Marco A. Morales A., Roger Pearce,
Shawna Thomas, Jyh-Ming Lien, Nancy M. Amato
{dawenx,marcom,rap2317,sthomas,neilien,amato}@cs.tamu.edu

Technical Report TR06-005
Parasol Lab.
Department of Computer Science
Texas A&M University

March 21, 2006

Abstract

Automatic motion planning has applications ranging from traditional robotics to computer-aided design to computational biology and chemistry. Probabilistic roadmap methods (PRMs) have been highly successful in solving many of these high degree of freedom problems. One important practical issue with PRMs is they do not provide an automated mechanism to determine what size roadmap to construct. Instead, users typically determine an appropriate roadmap size by trial and error and often construct larger maps than needed or build several maps before obtaining one that meets their needs. 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 independent processes, each of which generates samples and connections independently. IMG proceeds by adding these collections of samples and connections to an existing roadmap until it satisfies some specified evaluation criteria. We propose some general evaluation criteria and show how they can be used to construct different types of roadmaps, e.g., roadmaps that coarsely or more finely map the space. In addition to addressing the roadmap size question, the fact that each roadmap increment is independently and deterministically seeded has several other benefits such as supporting roadmap reproducibility, the adaptive selection of sampling methods in different roadmap increments, and parallelization. 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) [25, 28]. 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 [36].

For this reason, attention has focused on randomized approaches that sample and connect points in the robot's configuration space (C-space). Such methods include graph-based methods such as the *probabilistic roadmap methods* (PRMs) [27] (along with their various extensions and variants [1, 10–12, 23, 42]) for multiple query processing and tree-based methods such as Ariadne's Clew algorithm [32], RRT [29], and Hsu's expansive planner [24] for single query processing. PRMs 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 such as animation [6,30], deformable objects [4,7], computer-aided design/virtual prototyping [8,16], ligand docking [9,38], and protein folding [5,39].

However, the traditional PRM framework does not address several practical issues. One of the most important is that the PRM framework does not provide an automated way to build a roadmap of an appropriate size, i.e., large enough to adequately capture the connectivity of the free C-space and small enough to maintain efficiency. Determining the appropriate size is important – if the map is too large, then unnecessary computational resources (time and space) are used, and if it is too small, then if may falsely report that no path exists. In practice, users select a roadmap size they believe is appropriate, usually by trial and error, often resulting in larger maps than needed or in the construction of several maps before obtaining one that meets their needs. Instead of building a new roadmap, the user may expand the existing roadmap as in [27]. However, unless there is careful tracking of the random number generator seeds, results obtained in this manner are not reproducible, i.e., two roadmaps built using the same primitive operations and containing the same number of samples may differ. If this is the case, and if reproducibility is important, then the user must build an entirely new roadmap of larger size. For some applications such as protein folding where roadmaps can take days or weeks to build [40], this is impractical.

1.1 Our Contribution

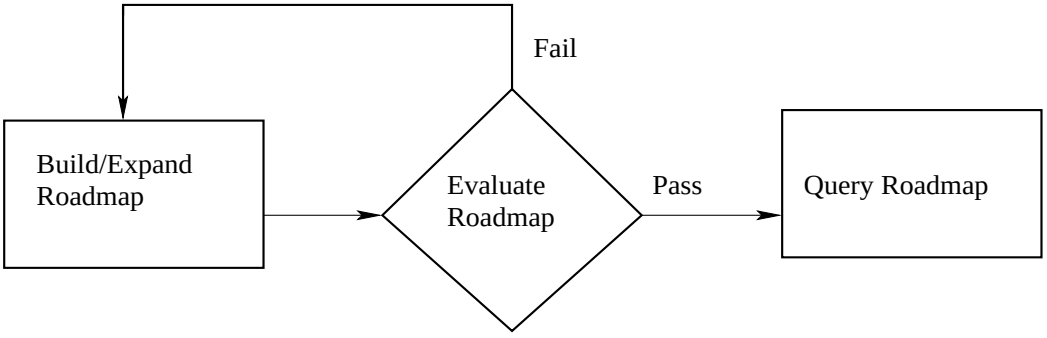

Figure 1: Flow diagram for Incremental Map Generation (IMG).

In this paper we propose a PRM-based framework called Incremental Map Generation (IMG) to automatically build a roadmap of an appropriate size for a given problem. This is implemented by iteratively building the roadmap until it satisfies a set of evaluation criteria (see Figure 1). We also propose evaluation criteria for different application domains.

This new framework differs from the traditional two-phase PRM method [27] in two important aspects. First, we include a new phase called "roadmap evaluation" which tests if the roadmap passes a set of evaluation criteria. If so, preprocessing ends and query processing begins. If not, preprocessing continues by expanding the roadmap by a fixed number of samples. This evaluation phase provides some measure of roadmap quality. Second, we

partition roadmap construction into several *independent sets*. Each computation set obtains a unique random seed and uses this seed to generate samples and connections. Note that IMG is *not* a new sampling method, instead it is a general strategy that can be applied to any sampling-based planner. There are several benefits provided by IMG:

- automatic determination of appropriate roadmap size,
- support for different roadmap construction strategies,
- roadmap reproducibility, and
- natural support for parallelization.

Automated determination of roadmap size. The most important feature of IMG is that it provides a framework to automatically determine the appropriate roadmap size for a given problem. The framework can accept a broad range of stopping or evaluation criteria which can be customized for particular applications or user preferences. In this paper, we propose evaluation criteria for different application domains.

Roadmap reproducibility. Our framework can support roadmap reproducibility by using a deterministic process to set the random number generator seed for each independent computation set. Note that this reproducibility remains valid even after more independent sets are added to an existing roadmap. Reproducibility is critical for applications such as protein folding where roadmaps can take days or even weeks to construct.

Roadmap construction strategies. New roadmap construction strategies can be easily introduced by adding another independent computation. For example, more powerful sampling strategies can be dynamically added to the map generation as more difficult areas, such as narrow passages, are identified. Thus, IMG provides a natural mechanism for adaptive planning.

Parallelization. The independent roadmap increments can easily be parallelized. Existing parallel implementations of randomized motion planners [3, 13–15, 31, 35, 40] do not provide solution consistency when varying the number of processors. Our framework can provide a consistent method for seeding random number generators, eliminating this processor count dependence.

2 Related work

The general PRM methodology [27] 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, that represents the free C-space. The query phase then connects any given start and goal configuration to the roadmap and returns a path if one exists in the roadmap.

The probability of failing to find a path from a probabilistic roadmap, given one exists, decreases exponentially as the number of samples in the roadmap increases [26]. However, it is difficult to decide beforehand the roadmap size required in practice. We propose a framework that automatically decides the roadmap size based on a set of user defined evaluation criteria. One difficulty is deciding appropriate evaluation criteria. In [20], the authors predefine a relevant query in each environment and continue building the roadmap until the query configurations are in the same connected component. This is helpful in environments where the user knows beforehand such a representative query. However, in more cluttered environments and in higher degree of freedom problems, defining such a query can be problematic.

In [21], coverage and maximal connectivity were used as an analysis tool to gain insight on sampling based methods. Coverage indicated how each query can be connected to the roadmap graph. 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 graph. The authors discretized C-space and determined for various techniques how long it takes before the free C-space has been adequately covered and connected. Unfortunately, discretizing C-space is not practical for high dof problems, and it is not always apparent when maximal connectivity has been reached.

A set of metrics are proposed in [33] to estimate how each new sample improves, or not, the planner's representation of C-space. With these metrics, the authors were able to identify three phases common to all sample-based planners: quick learning, model enhancement, and learning decay. They also demonstrated that the traditional scheme of testing a set of witness queries [19] can be misleading. The metrics in [33] classify the nodes added to

the model, in this case a roadmap, depending on their ability to increase the coverage and connectivity of the model.

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. Most importantly, this framework provides a systematic way to automatically decide how large a roadmap to build (see Figure 1). IMG is described in Algorithm 3.1. In the following sections we discuss two main aspects of our framework, incremental 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 all the criteria indicated by E.

1: **repeat**
2: Seed the RNG with s, the new seed for node set i.
3: Generate node set i (n nodes).
4: Add each node in node set i to roadmap R and perform connection.
5: **until** R meets criteria in E

3.1 Incremental Roadmap Construction

To build the map incrementally, we first divide roadmap construction into independent "sets" of size n, specified by the user. In order to ensure the independence of each set, we reseed the random number generator (RNG) for each set based on 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. This provides several key advantages:

- calculating the seed in a deterministic way based on a (possibly random) base seed supports reproducibility given the same base seed,
- adding independent roadmap components at run-time can allow map generation strategies to adapt as areas of C-space are characterized, and
- generating each set independently and deterministically facilitates parallelization.

For each set of generated nodes, we find "nearby" neighbors among all the nodes and perform connections. We continue constructing the roadmap until it meets all the evaluation criteria.

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. Partitioning the computation into independent pieces provides natural support for parallelization. In a parallel implementation, each process generates and connects an independent set of samples simultaneously. Then processes would communicate to integrate the independent sets together.

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 path-based evaluation. For roadmap progress evaluation, we propose to monitor the evolution of roadmap coverage and connectivity. For path based evaluation, we give two examples of evaluators needed for different applications. We study all evaluation methods and their performance in various situations.

3.2.1 Roadmap Progress Evaluation

In [33], every node is classified as it is inserted to the roadmap. 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) that exists 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. Note that it is relatively inexpensive to classify a node as cc-create or cc-merge. Based on the distribution of different types of nodes, it was observed that roadmaps map the C-space in three phases: quick learning, roadmap enhancement and learning decay.

Coverage and connectivity can be used as indicators of roadmap progress. As we construct the roadmap, we want to stop building a roadmap when we have achieved good coverage and connectivity at a reasonable cost. In the beginning of the roadmap construction, its coverage increases quickly and is easily detected, e.g., it can be characterized by a rate of change of all the node types higher than a user-defined parameter. What is more important and harder to detect is the expansion of components, i.e., the increase in connectivity. When a cc-expand node is added to a CC, the coverage of that CC increases and it also increases the probability that that component can be connected to other components. Also, when a cc-merge node is generated, the connectivity of the roadmap improves.

In this work, we use the "diameter" of the connected components as a measurement of component expansion. We define the *diameter* of a CC as the length of the longest shortest-path in the CC. We use *max-diameter* to represent the maximum diameter of all the CCs and *sum-diameter* to represent 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 coverage and connectivity of all the components in the roadmap. Component expansion can be approximated by the rates of change of max-diameter and sum-diameter. The diameter of a graph can be approximated and it is independent of the definition of distance [37].

In this evaluation method, we stop building a roadmap when the rate of change of max-diameter and sum-diameter over certain period of time is smaller than a user-defined threshold. More details are provided in Section 4

3.2.2 Path Based Evaluation

As mentioned earlier, the IMG framework can accept a broad range of stopping or evaluation criteria that can be customized for particular applications or user preferences. In this section, we give two examples of path based evaluation methods. These criteria can range from very simple (e.g., solving a user specified query) to complex (e.g., requiring the maximum network flow to exceed a threshold between two configurations).

Query Evaluation. This evaluator simply determines whether a roadmap can solve a user specified query. It attempts to connect the start and goal to the roadmap and returns successful if they are connected to the same connected component. This type of evaluator is useful when the user wants to solve a particular test problem.

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 protein structure transitions [5, 39]. To study how a protein changes from one target configuration to another, 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 threshold f.

4 Experimental Results

In this section, we show how IMG performs in practice.

4.1 Experimental Setup

All planners were implemented using the Parasol Lab motion planning library developed at Texas A&M University. For each problem, we build two versions of roadmaps, a tree and a graph. All results were run on 700MHz Intel PIII Xeon processors. We use RAPID [22] for collision detection calculations. For rigid-body motion planning,

two types of local planners, straight-line and rotate at 0.5 [2], are used to connect sampled configurations. For articulated linkage motion planning, we only use straight-line local planner.

A commonly used connection strategy called K-closest, tries to connect each node to its k "nearby" neighbors. K-closest does not distinguish success attempts from failure attempts. Nevertheless, identifying success and failures in connection attempts gives some indication of 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 effort on connections based on a node's local property, we use a modified version of K-Closest connection method called *L-Success-M-Failure* for connection. 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 failure attempts. We apply a 10-Success-20-Failure connection strategy for building tree roadmaps and 5-Success-20-Failure for building roadmaps with cycles (or graphs).

Recall that 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 of an appropriate size using different evaluation criteria. Our experiments used the following sampling methods:

- *Uniform random sampling*: samples are created by picking random values for each degree of freedom.
- *Gaussian-biased random sampling* [11]: sets of 2 samples are created, one is randomly sampled and the other is selected with a Gauss distribution among the samples that are at a distance d or less from the original sample. A collision-free sample is added to the roadmap when one is collision-free and the other is not,
- *Bridge-test random sampling* [23]: 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).

4.2 Automatically Building Roadmaps of Appropriate Size

We investigate the performance of the evaluator described in Section 3.2.1 in the four different environments shown in Figure 2: three rigid body problems and one free flying articulated linkage with 10 dofs.

We set the size of the independent sets as 50 samples and we compute the max-diameter among all CCs and the sum-diameter of all the CCs at the end of each independent set. The rate of change of the max-diameter ($PCMAX_i$) in the i^{th} set over its k previous sets is computed in the following way:

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

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

We stop roadmap construction when both $PCMAX_i$ and $PCSUM_i$ are smaller than a threshold τ, which is used to represent the desired improvement of the roadmap over a period of time.

Note that in the beginning of roadmap generation (the quick learning stage), there will be a large change in $PCMAX$ and $PCSUM$. Changes will drop when the enhancement stage starts.

4.2.1 A Case Study in Hook Environment

We now consider a case study for the hook environment using different planners. Figures 3-6 show results for both the tree and the graph roadmaps using the same random seed. All experiments used τ (0.0125). In each graph, the upper (lower) two plots correspond to the sum-diameter and max-diameter of the tree (graph) roadmaps,

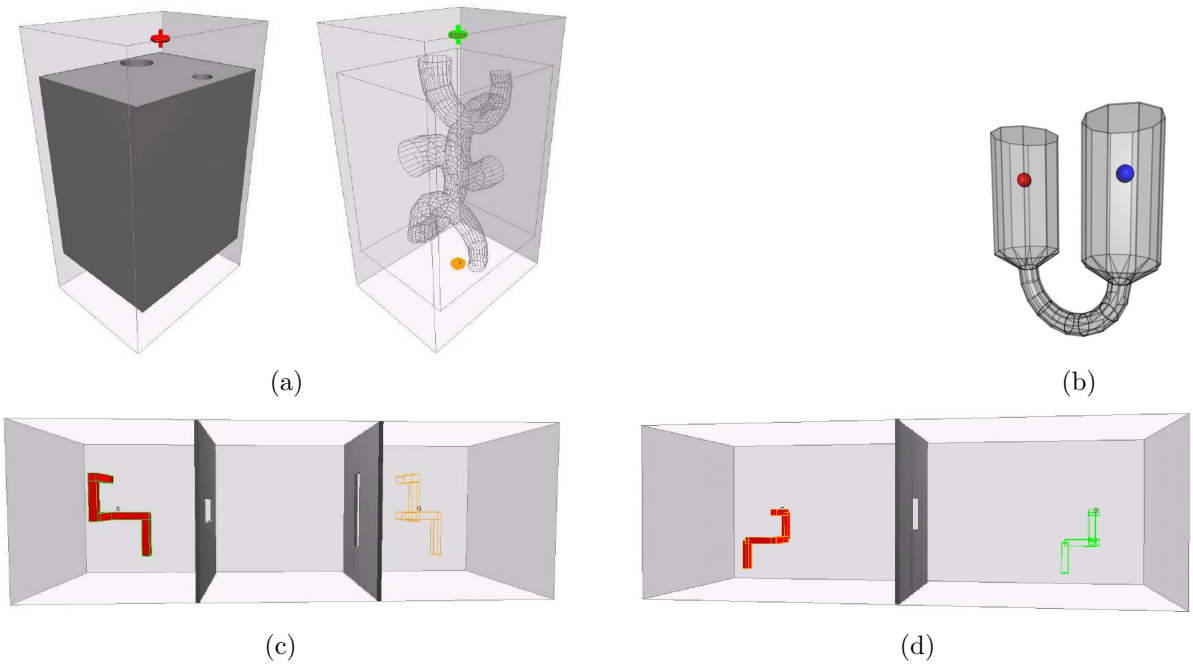

Figure 2: Problems studied. (a) Maze environment (solid/wire frame): rigid body robot must navigate the maze. (b) U shape environment: rigid body robot must navigate from one side to another through a narrow passage. (c) Hook environment: rigid body must rotate to move from one side of the wall to the other. (d) Hook manipulator environment: articulated linkage (10 dofs) must travel through the narrow hole to get from one end of the corridor to the other.

respectively. The vertical lines represent the stopping points that would be selected for different values of k (the number of sets of samples that the rate of change of the diameter is plotted over).

From the evolution of max-diameter and sum-diameter, it is clear that the roadmap grows rapidly in the beginning (the quick learning stage) 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, which also correlates with 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.

When using a fixed τ value, the general trend is that as the k value increases, the planner stops later because larger k values allow us to capture changes over larger periods. This is true for all the experiments we have run. This means that we can choose different values of k to decide how long we want to refine the roadmap. In another words, k can be used as a stopping criterion. Another way to use k is to find check points for a more powerful, and possibly more expensive, evaluation, or even to switch to another planner. It is also clear that for a given k value, different planners stop at different points. In particular, BasicPRM is the one which stops earliest. The intuition behind this is that BasicPRM is the slowest in terms of samples to progress, thus needing larger values of k to capture significant changes.

In the hook environment, we define a witness query from the first chamber to the last chamber and use the query evaluation as described in Section 3.2.1. The tree and graph roadmaps solve the query at the same point since we use the same random seed. It is clear from these figures that the max-diameter and sum-diameter are still growing after solving the predefined query. This confirms the observation in [33] that solving queries by itself is not enough to evaluate whether the planner is still making progress.

4.2.2 OBPRM in Several Environments

Here we show a case study for different environments using OBPRM. Figures 6, 7, 8, 9 shows results for both tree and graph roadmaps using the same random seed. We use a fixed value of k (10) in all these experiments

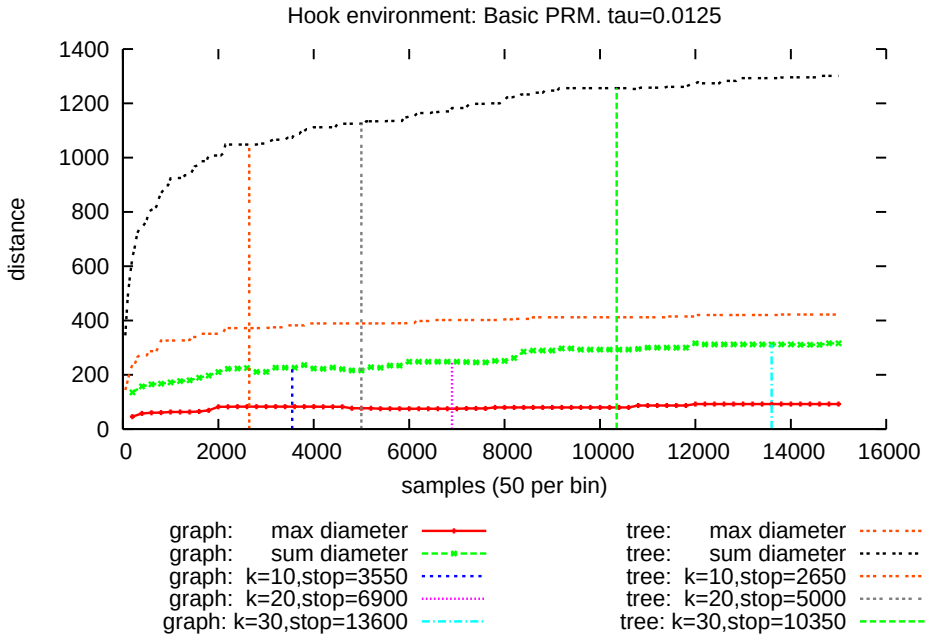

Figure 3: Hook environment, BasicPRM. The witness query was not solved when the roadmap has 15000 samples.

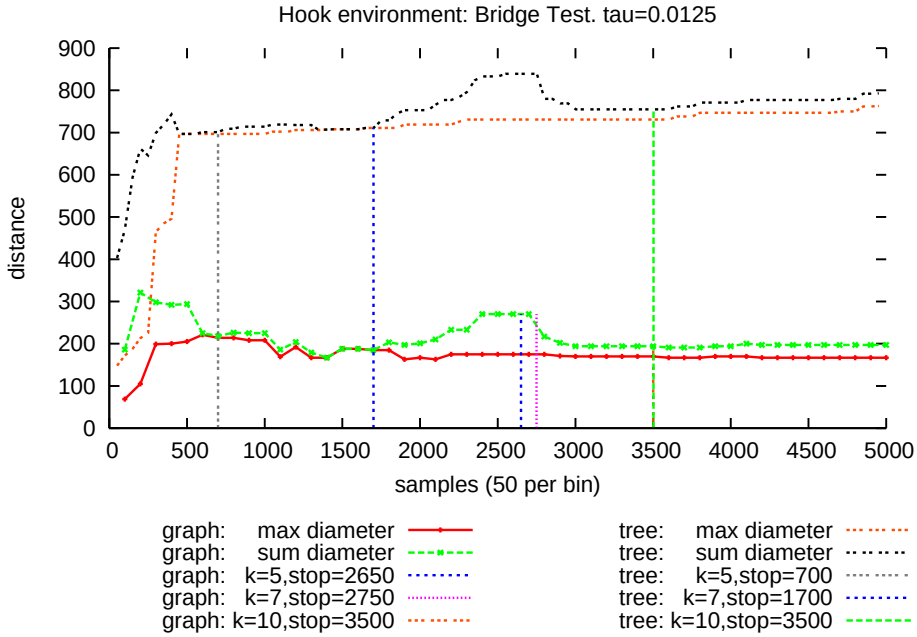

Figure 4: Hook environment, Bridge. The witness query was solved at 550 samples.

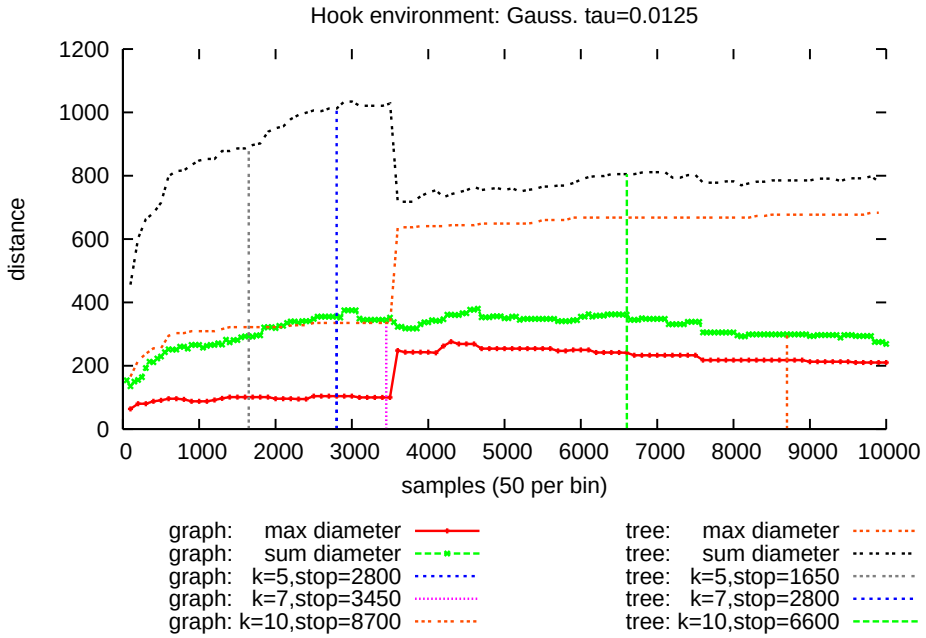

Figure 5: Hook environment, Gauss. The witness query was solved at 3600 samples.

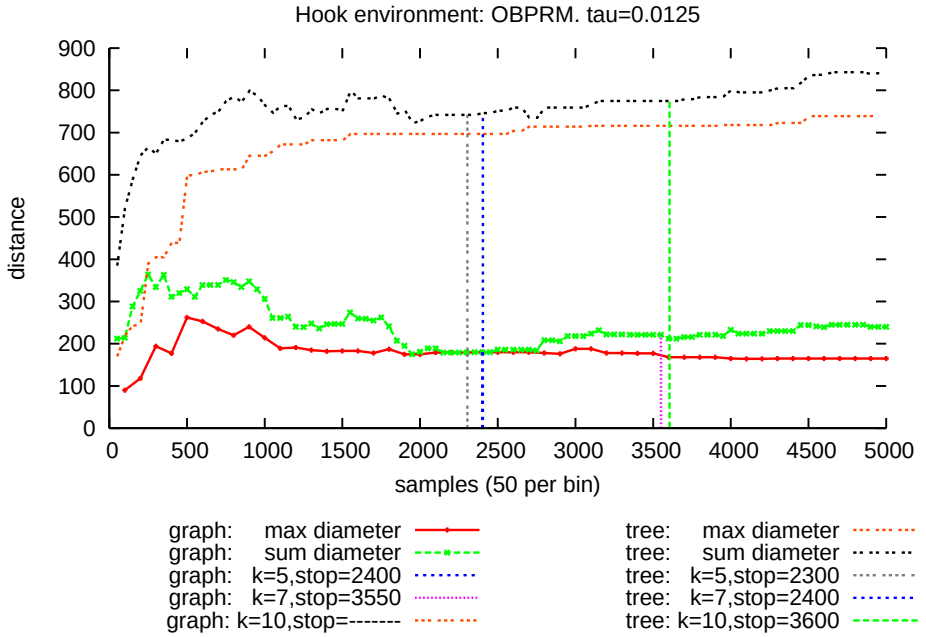

Figure 6: Hook environment, OBPRM. The witness query was solved at 500 samples.

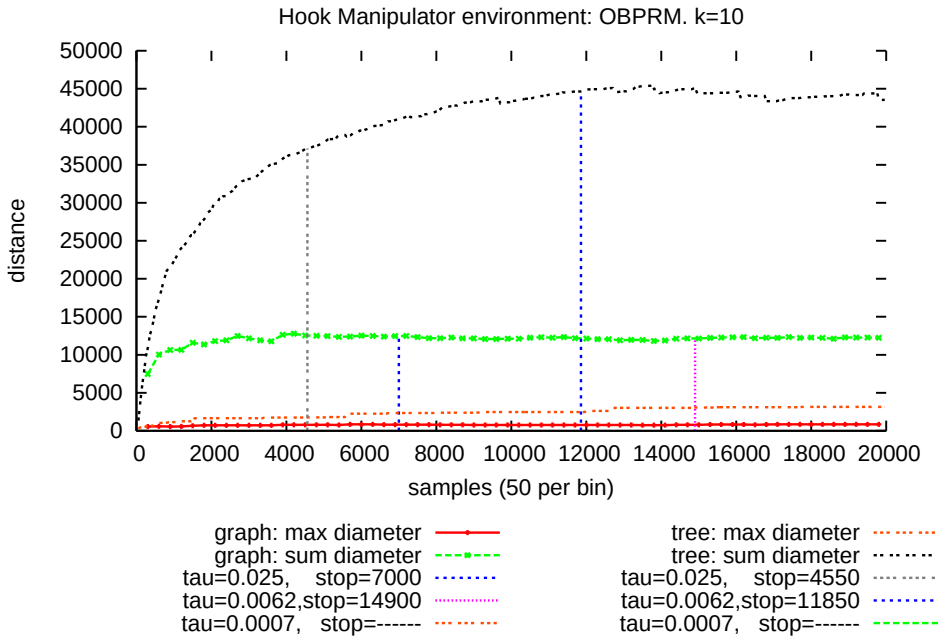

Figure 7: Hook manipulator environment, OBPRM. The witness query is not solved when the roadmap has 20000 samples.

and we plot the stopping point using different values of τ.

From our observations, k can be used to determine the desired speed of a planner's progress. By fixing $k = 10$ and varying τ we show the different stopping points for a desired rate of change of coverage and connectivity indicated through components' diameter. Different values of τ will stop roadmap construction at one of the three stages of roadmap construction: "quick learning, model enhancement, and learning decay."

In Figure 7 we observe that $\tau = 0.025$ roughly marks the end of the "quick learning" stage as it transitions into "model enhancement." Note, the predefined query was never solved using this roadmap, this can be seen also by $\tau = 0.0007$ which is not satisfied. In Figures 8 and 9 "learning decay" is clearly marked with $\tau = 0.007$.

In summary, τ is used to define the desired variability in coverage and connectivity indicated by the components' diameters, while k is set to define the minimum desired period for stabilization.

4.3 Application Specific Stopping Criteria

As discussed in Section 3.2, the IMG framework can accept a broad range of stopping criteria which 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 [17, 18, 41]. When calmodulin binds to Ca^{2+}, it undergoes a large-scale rearrangement [34] shown in Figure 10.

Using IMG, we were able to build a roadmap describing the C-space around both target configurations as well as the transition between them in 2 weeks of computation time. The traditional method would take at least 4 weeks of computation time to produce the same size roadmap.

5 Conclusion

In this paper, we proposed a framework to automatically determine an appropriate roadmap size for a given motion planning problem. This framework can accept a broad range of evaluation criteria which can be customized for particular applications. In addition to addressing the roadmap size issue, our framework provides other benefits of supporting roadmap reproducibility and different roadmap generation strategies, and easy parallelization. We

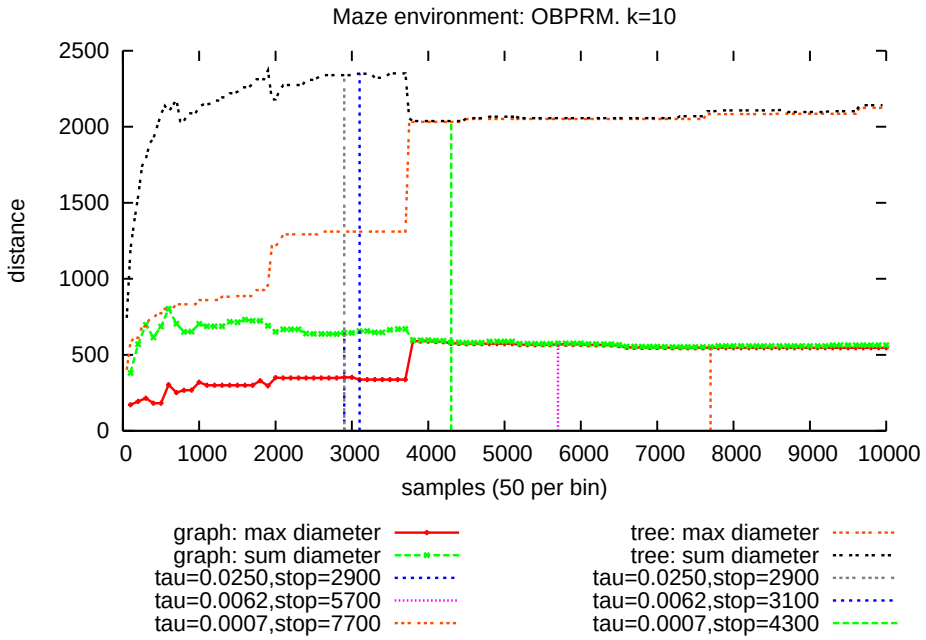

Figure 8: Maze environment, OBPRM. The witness query was solved at 3750 samples.

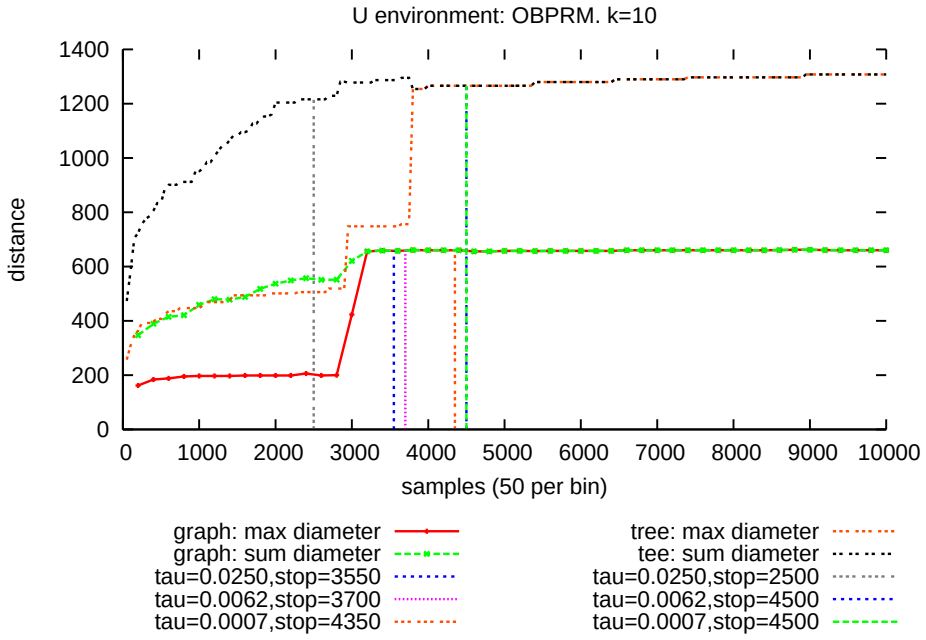

Figure 9: U environment, OBPRM. The witness query was solved at 2850 samples.

(a) (b)

Figure 10: Rearrangement of Calmodulin: (a) calcium-free state (1CFD) to (b) bound state (1CLL).

provide easy to define parameters that allow users to stop roadmap construction by satisfying criteria based on the quality of the roadmap. This has many potential applications that we plan to study.

In addition, there are several areas 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 energetic 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] N. M. Amato and L. K. Dale. Probabilistic roadmap methods are embarrassingly parallel. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 688–694, 1999.

[4] E. Anshelevich, S. Owens, F. Lamiraux, and L. Kavraki. Deformable volumes in path planning applications. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2290–2295, 2000.

[5] 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.

[6] O. B. Bayazit, J.-M. Lien, and N. M. Amato. Better group behaviors in complex environments using global roadmaps. In *Artif. Life*, pages 362–370, Dec 2002.

[7] O. B. Bayazit, J.-M. Lien, and N. M. Amato. Probabilistic roadmap motion planning for deformable objects. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2126–2133, May 2002.

[8] O. B. Bayazit, G. Song, and N. M. Amato. Enhancing randomized motion planners: Exploring with haptic hints. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 529–536, 2000.

[9] O. B. Bayazit, G. Song, and N. M. Amato. Ligand binding with OBPRM and haptic user input: Enhancing automatic motion planning with virtual touch. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 954–959, 2001. This work was also presented as a poster at *RECOMB 2001*.

[10] R. Bohlin and L. E. Kavraki. Path planning using Lazy PRM. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 521–528, 2000.

[11] 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.

[12] B. Burns and O. Brock. Sampling-based motion planning using predictive models. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2005.

[13] S. Carpin and E. Pagello. On parallel rrts for multi-robot systems. In *Proc. Italian Assoc. AI*, pages 834–841, 2002.

[14] D. Challou, D. Boley, M. Gini, and V. Kumar. A parallel formulation of informed randomized search for robot motion planning problems. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 709–714, 1995.

[15] D. J. Challou, M. Gini, and V. Kumar. Parallel search algorithms for robot motion planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 2, pages 46–51, 1993.

[16] H. Chang and T. Y. Li. Assembly maintainability study with motion planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1012–1019, 1995.

[17] A. Crivici and M. Ikura. Molecular and structural basis of target recognition by calmodulin. *Annu. Rev. Biophys. Biomol. Struct.*, 24:85–116, 1995.

[18] J. Evenas, A. Malmendal, and S. Forsen. Calcium. *Curr. Op. Chem. Biol.*, 2(2):293–302, 1998.

[19] R. Geraerts and M. H. Overmars. A comparative study of probabilistic roadmap planners. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, December 2002.

[20] R. Geraerts and M. H. Overmars. Sampling techniques for probabilistic roadmap planners. pages 600–609, 2004.

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

[22] 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.

[23] 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.

[24] 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.

[25] Y. K. Hwang and N. Ahuja. Gross motion planning – a survey. *ACM Computing Surveys*, 24(3):219–291, 1992.

[26] 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.

[27] 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.

[28] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[29] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 473–479, 1999.

[30] J.-M. Lien, O. B. Bayazit, R.-T. Sowell, S. Rodriguez, and N. M. Amato. Shepherding behaviors. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 4159–4164, April 2004.

[31] T. Lozano-Pérez and P. O'Donnell. Parallel robot motion planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1000–1007, 1991.

[32] 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.

[33] 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.

[34] 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.

[35] E. Plaku, K. E. Bekris, B. Y. Chen, A. M. Ladd, and L. E. Kavraki. Sampling-based roadmap of trees for parallel motion planning. *IEEE Trans. Robot. Automat.*, 2005.

[36] 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.

[37] R. Seidel. On the all-pairs-shortest-path problem. In *Proc. 24th Annu. ACM Sympos. Theory Comput.*, pages 745–749, 1992.

[38] A. Singh, J. Latombe, and D. Brutlag. A motion planning approach to flexible ligand binding. In *7th Int. Conf. on Intelligent Systems for Molecular Biology (ISMB)*, pages 252–261, 1999.

[39] 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.

[40] S. Thomas, G. Tanase, L. K. Dale, J. M. Moreira, L. Rauchwerger, and N. M. Amato. Parallel protein folding with STAPL. *Concurrency and Computation: Practice and Experience*, 17(14):1643–1656, 2005.

[41] H. Vogel. Calmodulin: a versatile calcium mediator protein. *Biochem. Cell Biol.*, 72(9-10):357–376, 1994.

[42] 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.