

# Metrics for Analyzing the Evolution of C-Space Models

Marco A. Morales A., Roger Pearce, Nancy M. Amato  
*Parasol Laboratory, Department of Computer Science*  
*Texas A&M University, College Station, Texas, 77843-3112, USA*  
{marcom,rap2317,amato}@cs.tamu.edu

**Abstract**—There are many sampling-based motion planning methods that model the connectivity of a robot’s configuration space (C-space) with a graph whose nodes are valid configurations and whose edges represent valid transitions between nodes. One of the biggest challenges faced by users of these methods is selecting the right planner for their problem. While researchers have tried to compare different planners, most accepted metrics for comparing planners are based on efficiency, e.g., number of collision detection calls or samples needed to solve a particular set of queries, and there is still a lack of useful and efficient quantitative metrics that can be used to measure the suitability of a planner for solving a problem. That is, although there is great interest in determining which planners should be used in which situations, there are still many questions we cannot answer about the relative performance of different planning methods. In this paper we make some progress towards this goal. We propose a metric that can be applied to each new sample considered by a sampling-based planner to characterize how that sample improves, or not, the planner’s current C-space model. This characterization requires only local information and can be computed quite efficiently, so that it can be applied to every sample. We show how this characterization can be used to analyze and compare how different planning strategies explore the configuration space. In particular, we show that it can be used to identify three phases that planners go through when building C-space models: quick learning (rapidly building a coarse model), model enhancement (refining the model), and learning decay (oversampling – most samples do not provide additional information). Hence, our work can also provide the basis for determining when a particular planning strategy has ‘converged’ on the best C-space model that it is capable of building.

## I. INTRODUCTION

In motion planning, a movable object (the robot) has to move in an environment while not violating motion constraints (such as reaching invalid configurations). While originating in robotics, motion planning has many applications beyond robotics including computer animation and games [1], [2] CAD [3], [4], and even computational Biology and Chemistry [5]–[9]. Since the motion planning problem is considered intractable [10]–[12], research on heuristic approaches has flourished [13]–[23]. Many of these heuristic planners make a model that approximates the space of valid robot configurations and motions. These approximations have enabled the solution of many previously unsolved problems. However, each heuristic has different strengths and weaknesses that make the quality of the models that they build dependent on the features of the problem being solved. While many

researchers have tried to compare different planners, most accepted metrics for comparing planners are based on computational efficiency, e.g., number of collision detection calls or samples needed to solve a particular set of queries, and generally only provide indirect information about how well the planner’s model represents the planning space. In this paper we are interested in this question, and in particular, in a specialization of it to monitor the incremental change, or evolution, of a planner’s C-space model.

Performance metrics that are commonly used in the analysis of planners include time and the number of basic operations needed to compute the model, and the amount of information about the planning space stored in the model. These metrics provide useful information, and they have been used in some studies to compare planners [24], [25]. However, they do not provide much information about the quality of the model produced. To enable a qualitative analysis of planners, we need metrics that help identify how well the models they build capture the important features of the planning space.

A qualitative property that has been used in planner analysis is called  $\epsilon$ -goodness [26]. This property shows that an environment can be modeled ‘easily’ if it is composed of samples that are  $\epsilon$ -good, meaning that they can be connected to a set of samples that covers at least  $\epsilon$  times the volume of the valid planning space. Unfortunately, this feature is not practical to compute for most interesting problems and there are many common problems which do not have this property.

In this paper we make some progress towards the goal of measuring how well a model represents the planning space. We identify a set of model features and relate them to the features of the underlying planning space, and propose a set of metrics that can be used to provide insight into how the model being constructed by a particular planning strategy evolves. Our proposed metrics can be measured locally and are efficient enough to be applied to each new sample considered by a sampling-based planner to characterize how that sample improves, or not, the planner’s current C-space model. Accordingly, an important application of our work will be to provide a stopping criterion for sampling-based planners because it can be used to determine when a particular planning strategy has ‘converged’ on the best C-space model that it is capable of building.

We provide results that illustrate how our metrics can be used to analyze and compare how different planning strategies explore the configuration space. We show how they can be used to identify three phases that planners go through when building C-space models: quick learning (rapidly building a coarse model), model enhancement (refining the model),

<sup>1</sup>This research supported in part by NSF Grants EIA-0103742, ACR-0081510, ACR-0113971, CCR-0113974, ACI-0326350, and by the DOE. Morales supported in part by a Fulbright/Garcia Robles (CONACYT) Fellowship.

and learning decay (oversampling – most new samples do not provide additional information). We also show that this characterization can capture planner differences that are not identified using previous techniques, such as a method’s ability to solve a particular set of motion planning queries.

## II. C-SPACE

A configuration is a description of the placement of all points of the robot with respect to a coordinate system. We can represent a configuration  $q$  with  $d$  parameters, or degrees of freedom, each corresponding to a unique component of the robot (e.g., object positions and orientations, link angles and displacements, etc.). Thus, each configuration is a point  $q = (x_1, \dots, x_d)$  in the  $d$ -dimensional configuration space (C-space)  $\mathcal{C}$  consisting of all possible robot configurations in the given environment.

We define a boolean function  $valid(q)$  that is true if  $q$  is a valid configuration for the robot in its environment, and false if it is not. The subset of valid configurations in  $\mathcal{C}$  is the free space (C-free)  $\mathcal{F}$ .

For two configurations  $q$  and  $q'$ , we define a boolean function  $visible(q, q')$  that is true if a specified method can find a *path* or continuous sequence of adjacent (at a required resolution) configurations  $\{q_1, q_2, \dots, q_n\}$  where  $q_1 = q$ ,  $q_n = q'$ , and  $valid(q_i) = true$ ,  $1 \leq i \leq n$ . For example, the straight line planner will determine that  $q$  can see  $q'$  if the straight line between  $q$  and  $q'$  is composed of only valid configurations. Note that visibility is not necessary symmetric – this is determined by the method used to define visibility.

Connectability is related to visibility. For two configurations  $q$  and  $q'$ , we define a boolean function  $connectable(q, q')$  that is true if there exists a *path* or continuous sequence of configurations  $\{q_1, q_2, \dots, q_n\}$  where  $q_1 = q$ ,  $q_n = q'$ , and  $visible(q_i, q_{i+1}) = true$ ,  $1 \leq i < n$ .

### A. C-space properties

We define a connected component  $\mathcal{CC}$  of the free C-space as a maximal subset of configurations  $\mathcal{CC} \subseteq \mathcal{F}$  such that  $\forall q, q' \in \mathcal{CC}, connectable(q, q') = true$ . The free configuration space  $\mathcal{F}$  may be formed by one or more connected components.

1) *Coverage*: Let  $q$  be a configuration in  $\mathcal{F}$ . We define the *coverage* of  $q$  as the subset of  $\mathcal{F}$  that is visible from  $q$ :

$$Cov(q) = \{\forall q' \in \mathcal{F} | visible(q, q') = true\} \quad (1)$$

We define the *coverage* of a set of configurations  $Q = \{q_1, \dots, q_n\}$  as the union of the coverage of its elements:

$$Cov(Q) = \bigcup_{i=1}^n Cov(q_i) \quad (2)$$

2) *Connectivity*: One common operation when modeling the connectivity of the C-free  $\mathcal{F}$  is to define connections between a pair of samples  $(q, q')$ . Two samples  $(q, q')$  can be connected when  $visible(q, q') = true$

Let us define the *connectivity region* of a connected subset of samples  $\mathcal{CC}$  as  $Cov(\mathcal{CC})$

## III. C-SPACE MODELING

We define a model  $M$  of C-space as a subset  $V$  of configurations selected from configurations in  $\mathcal{F}$  and a subset  $E$  of pairs of configurations selected from all the visible configuration pairs  $V \times V$ . This definition is general enough to cover the models constructed by all sampling-based planners of which we are aware, e.g., graph-based planners such as, e.g., PRM [14], and tree-based planners such as, e.g., Ariadne’s Clew [21], RRT [22], or Hsu’s method [23].

The configurations in  $V$ , called *nodes* or *vertices*, are the configurations that are selected (sampled) by the particular planner. While there are exceptions, in this paper we will assume that it is known that  $valid(q) = true, \forall q \in V$  (these concepts can be generalized to cover those other cases as well).

Let  $lp$  be the function (typically called a local planner) used to test the visibility of two configurations  $q$  and  $q'$ . The function  $lp$  tests a path or sequence of adjacent (at a required resolution) configurations  $p = \{q_1, \dots, q_n\}$  by checking  $valid(q_i), 1 < i < n$ . Let  $visible_{lp}$  be the visibility function defined by  $lp$ . The pairs in  $E$ , called *edges*, are the node pairs  $(q, q')$  that are selected according to some strategy (e.g., a strategy commonly used in PRM planners is  $k$ -closest, which selects the  $k$  closest nearest neighbors for each node in  $V$ ). As with the nodes, in this paper we will assume that  $visible_{lp}(q, q') = true, \forall (q, q') \in E$  (again, these concepts can be generalized to cover other cases as well).

### A. Model properties

A C-space model  $M$  should reflect two features of C-space: coverage and connectivity.

1) *Coverage*: A model  $M$  has *perfect coverage* if every configuration in C-free can see at least one node of the model.

2) *Connectivity*: We define a path between two nodes  $v$  and  $v'$  as a sequence of nodes  $p = \{v_1, \dots, v_n\}$  where  $v_1 = v$  and  $v_n = v'$  and  $(v_i, v_{i+1}) \in E$  for  $1 \leq i < n$ . A connected component in the model is a set  $CC_m \subseteq V$ , so that  $\forall v, v' \in V$  there is a path between  $v$  and  $v'$ . A model  $M$  has *perfect connectivity* when it has perfect coverage and for every pair  $(p, p') \in \mathcal{F}$  such that  $connectable(p, p') = true$ , there is a path in  $M$  between vertices  $v$  and  $v'$ , where  $visible(p, v) = visible(p', v') = true$ .

3) *Efficiency*: An additional desired feature of the model is to have a minimum number of nodes and edges, while still achieving the best possible coverage and connectivity.

### B. Effects of sampling configurations and connections on coverage, connectivity, and efficiency

Given a model  $M$ , a planner adds a valid sampled configuration  $v$  and a selected subset of all its valid connections producing the model  $M'$ . This operation changes the connectivity and coverage of the original model  $M$  in exactly one of the following ways:

- 1) **cc-create** —  $v$  falls outside of the coverage of all the components in  $M$ . As a consequence, a new component  $CC$  with  $v$  as its only node is created.  $Cov(M')$  increases by the coverage of  $v$  and the connectivity of the components already in  $M$  *remains constant*, but the connectivity of  $M'$  improves due to the new component. (See Fig. 1(b,c).)

- 2) **cc-merge** —  $v$  falls inside the coverage region of more than one component of  $M$ . As a consequence, the components and their coverage regions merge, reducing the number of components.  $Cov(M)$  may remain constant but connectivity of  $M$  improves. (See Fig. 2(a).)
- 3) **cc-expand** —  $v$  falls inside the coverage of exactly one component  $CC$  of  $M$ , but  $Cov(v)$  is not a subset of  $Cov(CC)$ .  $Cov(M)$  increases but connectivity of  $M$  remains constant. (See Fig. 2(b).)
- 4) **cc-oversample** —  $v$  falls inside the coverage of exactly one component  $CC$  in  $M$  and coverage and connectivity of  $M$  remain constant. (See Fig. 2(c).)

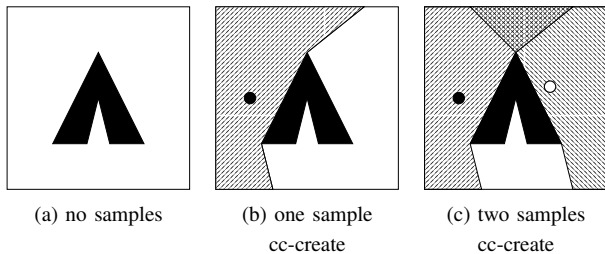


Fig. 1. (a) A two-dimensional environment with a polygonal obstacle. (b) One sample is added. (c) One more sample is added. Note that the coverage of the two samples overlaps above the obstacle.

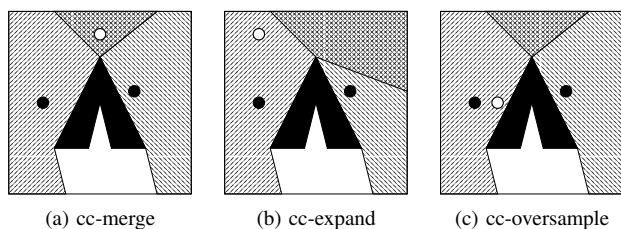


Fig. 2. New samples (hollow dots) are added. (a) A cc-merge sample falls in the intersecting coverage of the two existing samples connecting the two regions. (b) A cc-expand sample increases the coverage of the component to the left, but does not change the connectivity. (c) Oversampling does not increase coverage or connectivity.

Also, the new nodes and edges in  $M'$  can be classified as efficient or inefficient. A model  $M$  is efficient if it only contains efficient nodes and efficient edges. A node is efficient when it increases the knowledge of the model about the C-space. So, a cc-oversample node does not improve  $M$  and is inefficient. An edge is efficient when it is necessary to maintain the connectivity of its component. One indication of an inefficient edge is that it produces a redundant path between a pair of nodes (e.g., a cycle).

### C. Metrics to characterize C-space Models

To characterize a C-space model  $M$ , we propose to classify every new node according to how it changes the new model  $M'$  with respect to  $M$  based on the cases described above. This way, each node is classified as being a *cc-create*, *cc-merge*, *cc-expand*, or *cc-oversample* node. The new edges in  $M'$  can also be classified as efficient or inefficient.

The distributions of the types of nodes as the model construction progresses should be correlated with the planner's ability to increase its knowledge about the planning space.

For example, a situation where the ratio of new oversampling nodes keeps growing as the others stall indicates that the planning strategy might have reached the limit of its ability to improve the model.

The specific mechanism to detect each type of node and edge will depend on the desired accuracy of the measurements and the cost that can be incurred in their extraction.

## IV. APPLICATION: ANALYSIS OF NODE SAMPLING STRATEGIES IN PRM ROADMAPS

Any planner producing C-space models that fit the definitions of Section III can be studied with the metrics defined above. We demonstrate their use on several variants of the Probabilistic Roadmap Method (PRM) [14]–[16]. PRMs model the C-space of a given problem with a graph, also called a roadmap. Traditionally, PRM roadmaps are made in two main steps: node generation and node connection. During node generation, robot configurations are randomly sampled from the C-space and tested for validity—in typical robotic applications through a collision detection test. Valid samples are kept as roadmap nodes. Then, during node connection, pairs of nearby configurations (as determined by a selected distance metric [27]) are selected as candidates for connection, which is then tested by deterministic local planners. The pairs that can be connected are stored as edges in the roadmap. The same roadmap can be applied to solve multiple queries requiring the robot to move between a pair of configurations.

Many strategies for node generation and node connection have been developed to try to produce better roadmaps at a lower cost. By extracting the metrics proposed above to different sampling and connection strategies, we can gain insight about the best use of each strategy. To study the effectiveness of the metrics proposed, we analyze several node generation methods and the incremental construction of roadmaps that are built using them. We study both cumulative and incremental measures to try to understand and distinguish the progress made using a particular node generation method toward *convergence*.

### A. PRM planners

In the experiments, we use the PRM methods and parameters described below. These parameters were set in an attempt to treat all node sampling methods tested equally and to enable the comparison to concentrate on the nodes sampled.

- **Problem domain** The problems studied involve robots that are (possibly articulated) rigid bodies moving in three-dimensional environments.
- **Sampling strategies** We applied five sampling methods to each environment: BasicPRM [14], Bridge-Test [28], GaussPRM [18], OBPRM [17], and MAPRM [19]. These methods were chosen to compare biased vs. unbiased sampling methods as well as the effects of using local C-Space information to guide sampling.
- **Valid node** — A node is tested for validity by placing the robot in the node configuration and testing for collision.
- **Connection pair selection** — Connections were attempted between each node added to the roadmap and the  $k$ -closest ( $k = 10$ ) nodes already there.

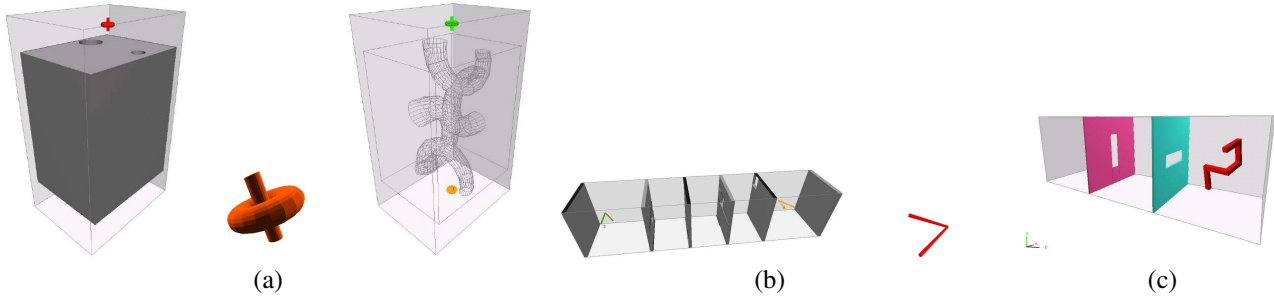


Fig. 3. (a) *maze*: solid view, 6-DOF robot, and wire view. (b) *walls*: environment and serial robot (7 DOF). (c) *hook*.

- **Node visibility (local planner)** — Two nodes are *visible* if they can be connected by either the straight-line or the rotate-at-s ( $s=0.5$ ) local planners [27].

### B. Environments

We applied the five node generation methods to three different environments.

The *maze* environment is composed of a series of tunnels, some of them are dead ends. The robot is a rigid body with 6 DOFs that has to go from the top to the bottom part of the maze. (Figure 3(a))

The *serial walls* environment is composed of five chambers divided by walls with small holes in them. The robot is an articulated 2-link manipulator with 7 DOFs. (Figure 3(b)).

The *hook* environment is composed of two walls with narrow holes and the robot is a 6-DOF rigid body that can only traverse the narrow passages using translational and rotational motion. (Figure 3(c)).

### C. Experiments

For each sampling method, nodes were added to the roadmap in an iterative fashion in ten independent runs using different seeds for the random number generator. Section V discusses the results obtained.

**Node classification.** We extracted metrics after each sample was added and connected to the roadmap. In particular, each node was classified as one of the four types described in Section III-B. A node that cannot be connected to an existing roadmap component is a *cc-create* node. A node that causes a reduction of the number of components in the roadmap is a *cc-merge* node. A node  $v$  that connects to another node  $v'$ , but cannot be connected to all of  $v'$ 's neighbors, is a *cc-expand* node. Note that this is a conservative heuristic, that identifies some, but not all, cc expand nodes. And finally, a node that does not fall in any of the previous categories is an *cc-oversample* node.

**Witness queries.** One technique that has been used in the past to evaluate different planners, is to develop a set of queries (witness queries) and to compare methods based on how many of those queries they can solve [24]. This has also been used as a ‘convergence’ test to determine when a sufficient roadmap has been constructed. This kind of test is expensive to compute and, as we show in our experiments, can also be misleading. Indeed, we will see that our proposed node type metrics can be used as better criteria for convergence tests.

## V. DISCUSSION OF RESULTS

The metrics extracted during roadmap construction let us monitor the progress made by the planner in the exploration of C-space. To view the planners’ abilities to map C-space we obtained two types of plots using our metrics.

- **Cumulative plots.** The fraction of nodes in each category (cc-create, cc-merge, cc-expand, cc-oversample) are plotted against the cumulative number of samples. The horizontal axis is shown in logarithmic scale to amplify the resolution for early samples (Figure 4).
- **Temporal histograms.** This set of results analyzes the distribution of samples over time. The samples are temporally partitioned into equal-sized bins and the relative percentage of each type of node for each bin is shown in a stacked histogram (Figure 5).

In all plots, we also show the percentage of random queries (witness queries) that can be solved with the roadmap at that time. Sharp changes in these results can indicate that important connections were made in the roadmap, e.g., between chambers. We will also compare witness queries with our metrics for measuring roadmap quality.

### A. Metrics as indicators of sampling progress

In all our experiments we observed similar trends in the evolution of the roadmaps as measured by our proposed metrics. Three representative examples are shown in Figure 4; more results can be found in [29]. In particular, our metrics identified three stages of roadmap evolution for all planners:

- 1) **quick learning** — the cc-create nodes start off high and quickly decline. Simultaneously, cc-merge nodes start appearing and cc-expand nodes start a continuous growth. In this stage, the roadmap quickly improves coverage and connectivity.
- 2) **roadmap enhancement** — cc-merge nodes drop while cc-expand nodes stabilize. Here, cc-expand nodes may help connect areas that are hard to reach.
- 3) **learning decay** — cc-expand nodes start declining and most new nodes are cc-oversample nodes. Although the planner is still exploring previously uncovered areas, it does so at a slower rate. The absence of new cc-merge nodes and a declining number of cc-expand nodes corresponds to a reduced probability of improving roadmap connectivity.

Despite the similarities in the trends of all the planning strategies, we do find some important differences. For example, the rate at which cc-create and cc-merge nodes decline in

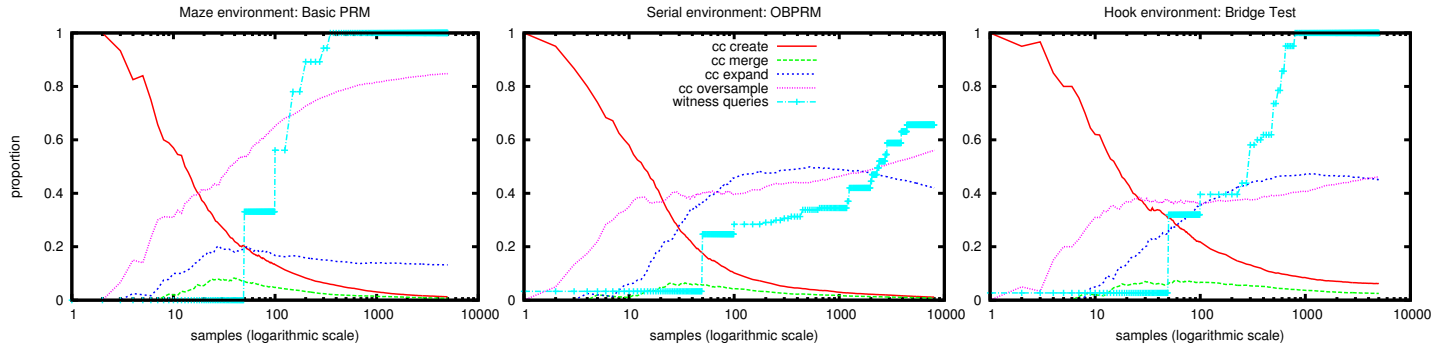


Fig. 4. Proportion of samples of each type as nodes are added to the roadmap in three environments. (left) maze environment, Basic PRM sampling. (center) serial environment, OBPRM sampling. (right) hook environment, Bridge Test sampling.  $k$ -closest connection ( $k = 10$ ) in all cases.

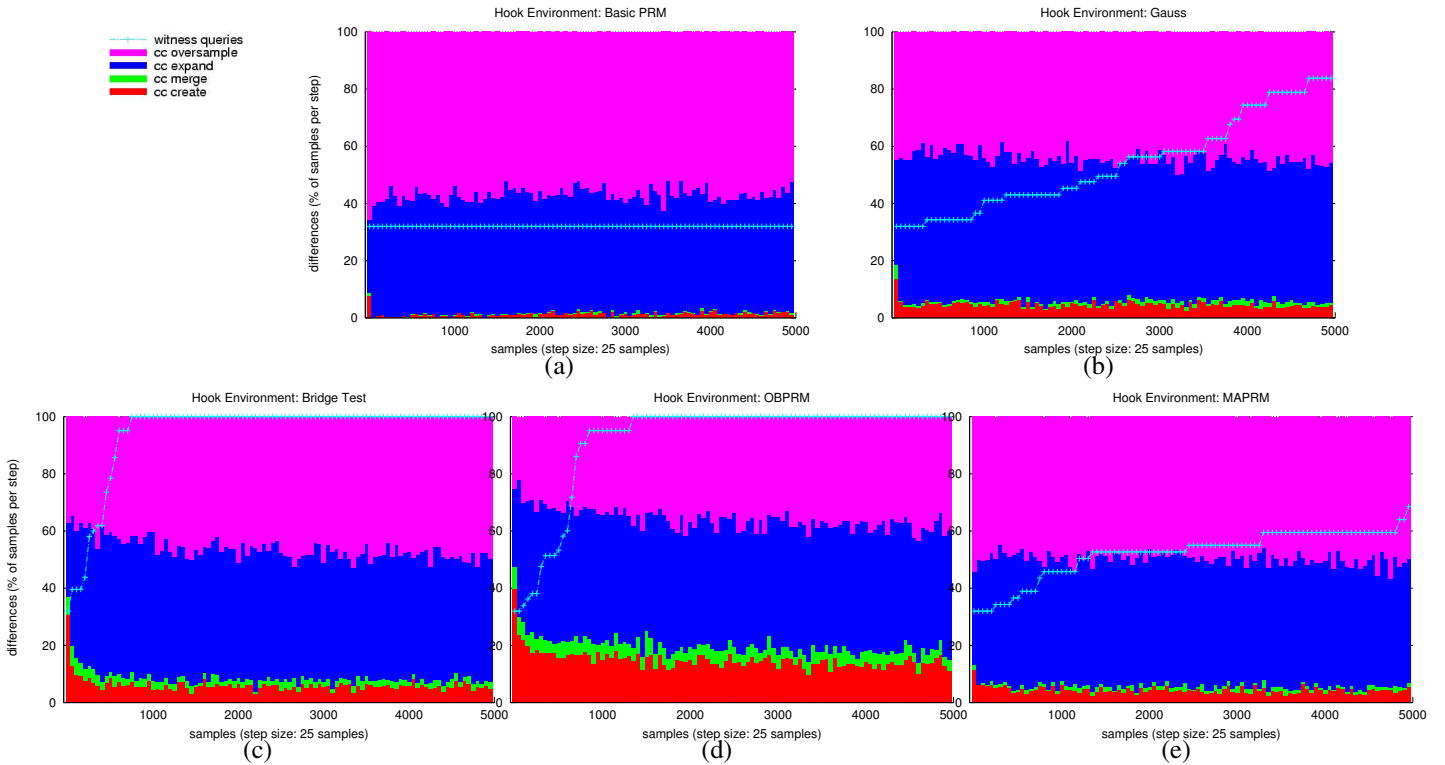


Fig. 5. Nodes of each type in groups of new samples added to the *hook* environment. Sampling methods: (a) Basic PRM, (b) Gauss, (c) Bridge Test, (d) OBPRM, (e) MAPRM.  $k$ -closest connection ( $k = 10$ ) in all cases

OBPRM is slower than other methods, and its *cc-expand* nodes reach higher proportions. In this last aspect, Gauss PRM ranks second. These differences can be noted in the rate of change of each node type in the temporal histograms in Figure 5. There is one plot for each of the planning strategies studied. We see that in all cases there is an exponential decay of *cc-create* nodes, and, while there are some variations, the *cc-merge* set is always the smallest. Basic PRM, which distributes samples uniformly in the space, can mainly produce expanding nodes. Gauss PRM, which selects samples that are close to invalid samples (based on uniform sampling of pairs of configurations), is better at producing *cc-create* nodes, but this drops off rapidly. The Bridge Test planner, that places nodes in free areas that are between two invalid samples chosen from a uniform distribution, has a slower drop of *cc-create* nodes, and

makes more *cc-merge* nodes. OBPRM, which pushes invalid samples away from C-obstacles to find samples close to the boundary, has the highest percentage of *cc-create* nodes and keeps producing a significant number throughout. MAPRM, which pushes valid and invalid configurations toward the medial axis of the valid space, keeps generating *cc-create* and *cc-merge* nodes. We can also note that in all methods, except from Basic PRM, the *cc-oversample* nodes increase slowly. This trend should continue until the planner has constructed the best roadmap that it can.

In all the plots, we also show the number of witness queries that can be solved with the current roadmap. These results illustrate how using the ability to solve witness queries to measure roadmap quality may be misleading. For example, by simply comparing the witness query curves in Figure 5,

OBPRM and Bridge Test appear to be similar and so do GaussPRM and MAPRM. However, we see that our proposed metrics show that there is in fact quite a difference in the progress these planners make in modeling the planning space.

Finally, we computed the average cost of adding a node to the roadmap with each strategy. Figure 6 shows the accumulated costs of the samples (and their connections) for each planning strategy in a representative example. We see that Bridge Test nodes were the most expensive, followed by OBPRM, MAPRM, Gauss, and Basic PRM. This cost should be taken into account when deciding which strategy should be used in each of the three stages of roadmap construction identified above.

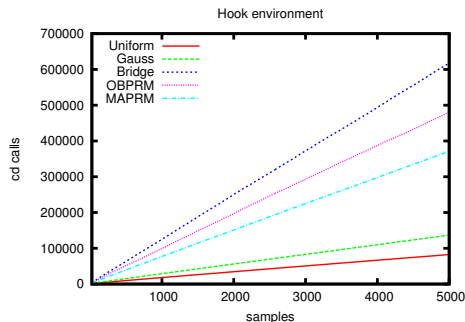


Fig. 6. Collision detection calls in the hook environment for different methods.

## VI. CONCLUSION

In this paper we propose a method that can be used to characterize each sample considered by sampling-based motion planning methods. We showed that this characterization can be used to analyze the performance of a planner and to identify three learning phases in the C-space model evolution. This approach is suitable to be used in an on-line fashion and can be used as the basis of a convergence test to determine when roadmap construction strategies should be modified or halted. In future work, we plan to use these metrics on-line to adaptively modify a planner's characteristics.

## REFERENCES

- [1] 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)*, April 2004, pp. 4159–4164.
- [2] J.-M. Lien, S. Rodriguez, J.-P. Malric, and N. M. Amato, "Shepherding behaviors with multiple shepherds," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, April 2005, pp. 3413–3418.
- [3] 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)*, 2000, pp. 529–536.
- [4] H. Chang and T. Y. Li, "Assembly maintainability study with motion planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1995, pp. 1012–1019.
- [5] N. M. Amato and G. Song, "Using motion planning to study protein folding pathways," *J. Comput. Biol.*, vol. 9, no. 2, pp. 149–168, 2002, special issue of *Int. Conf. Comput. Molecular Biology (RECOMB)* 2001.
- [6] 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)*, 2001, pp. 954–959, this work was also presented as a poster at *RECOMB* 2001.
- [7] X. Tang, B. Kirkpatrick, S. Thomas, G. Song, and N. M. Amato, "Using motion planning to study rna folding kinetics," *J. Comput. Biol.*, vol. 12, no. 6, pp. 862–881, 2005.
- [8] S. Thomas, X. Tang, L. Tapia, and N. M. Amato, "Simulating protein motions with rigidity analysis," in *Proc. Int. Conf. Comput. Molecular Biology (RECOMB)*, 2006.
- [9] 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)*, 1999, pp. 252–261.
- [10] J. H. Reif, "Complexity of the mover's problem and generalizations," in *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, San Juan, Puerto Rico, October 1979, pp. 421–427.
- [11] J. T. Schwartz and M. Sharir, "On the 'piano movers' problem II: General techniques for computing topological properties of real algebraic manifolds," *Adv. Appl. Math.*, vol. 4, pp. 298–351, 1983.
- [12] J. F. Canny, *The Complexity of Robot Motion Planning*. Cambridge, MA: MIT Press, 1988.
- [13] J. Barraquand and J. C. Latombe, "Robot motion planning: A distributed representation approach," *Int. J. Robot. Res.*, vol. 10, no. 6, pp. 628–649, 1991.
- [14] 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.*, vol. 12, no. 4, pp. 566–580, August 1996.
- [15] M. Overmars, "A random approach to path planning," Computer Science, Utrecht University, The Netherlands, Tech. Rep. RUU-CS-92-32, 1992.
- [16] M. Overmars and P. Svestka, "A probabilistic learning approach to motion planning," in *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 1994, pp. 19–37.
- [17] 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*. Natick, MA: A.K. Peters, 1998, pp. 155–168, proceedings of the Third Workshop on the Algorithmic Foundations of Robotics (WAFR), Houston, TX, 1998.
- [18] 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)*, vol. 2, 1999, pp. 1018–1023.
- [19] 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)*, vol. 2, 1999, pp. 1024–1031.
- [20] J.-P. Laumond and T. Siméon, "Notes on visibility roadmaps and path planning," in *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2000.
- [21] P. Bessiere, J. M. Ahuactzin, E. G. Talbi, and E. Mazer, "The Ariadne's clew algorithm: Global planning with local methods," in *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, vol. 2, 1993, pp. 1373–1380.
- [22] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1999, pp. 473–479.
- [23] D. Hsu, R. Kindel, J. C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," in *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2000, pp. SA1–SA18.
- [24] 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.
- [25] S. M. LaValle and M. S. Branicky, "On the relationship between classical grid search and probabilistic roadmaps," in *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, December 2002.
- [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)*, May 1995, pp. 353–362.
- [27] 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.*, vol. 16, no. 4, pp. 442–447, August 2000, preliminary version appeared in *ICRA* 1998, pp. 630–637.
- [28] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "Bridge test for sampling narrow passages with probabilistic roadmap planners," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2003, pp. 4420–4426.
- [29] M. A. Morales A., R. Pearce, and N. M. Amato, "Metrics for analyzing the evolution of c-space models," Parasol Lab, Dept. of Computer Science, Texas A&M University, Tech. Rep. 06-002, February 2006.