Proceedings of the 2nd International Conference of Control, Dynamic Systems, and Robotics
Ottawa, Ontario, Canada, May 7-8 2015
Paper No. 173

Probabilistic Roadmap Planner with Adaptive Sampling Based on Clustering

M. Safilian
Amirkabir University of Technology
Tehran, Iran
m.safilian@aut.ac.ir

S. Eghbali
University of Waterloo
Waterloo, Ontario, Canada
sepehr.eghbali@uwaterloo.ca

A. Safilian
McMaster University
Hamilton, Ontario, Canada
safiliaa@macmaster.ca

A. Nouri
Carleton University
Ottawa, Ontario, Canada
arashnourisagharlou@cmail.carleton.ca

Abstract- Path planning problem is a widely applied yet complex problem in robotics. It aims to find the shortest collision-free path in a given environment for the robot's movement from an initial to a target configuration. One popular technique to solve it is Probabilistic Roadmap Planner (PRM), which samples the environment to form an approximate representation of the problem space. Using uniform sampling can reduce the efficiency of this method since not all the regions of environment are similar in terms of their complexity. In this paper, we propose a novel approach to address this problem, which includes two steps. First, the problem environment is clustered such that more complex regions are separated form simple ones. In the second step, the sampling is performed which is more biased towards the clusters which contain complex regions. Our experimental results indicate that using the notion of clustering can significantly boost the performance of PRM techniques, specially in complex environments.

Keywords: path planning, PRM, clustering, sampling

1 Introduction

Robotic path planning has received significant amount of attention recently due to its broad applications. Informally speaking, the goal of this problem is to find a path for a robot from an initial position to a final position while avoiding collision between the robot and the obstacles (Choset et al., 2005). The path planning problem is formally defined using: (1) C as the space of all possible configurations of the robot (2) C_{free} as the set of configurations not colliding with obstacles of environment (3) $C \setminus C_{free}$ as the obstacle space (4)

c_{init}, c_{goal} is the starting and goal configuration. The goal of path planning is to find an efficient path (set of configurations) in C_{free} from the c_{init} to c_{goal}.

The path planning problems is an interesting yet hard-to-solve problem; in general the path planning problem is PSPACE- complete (Canny et al., 1988). As a result, solving this problem is usually intractable in high dimensional spaces. Nonetheless, due to its broad applications, many heuristics have been proposed recently to solve path planning problem in complex environments. Most of these methods can be categorized into three groups: cell decomposition methods, potential field methods and sampling based methods (Latombe et al., 1991).

Probabilistic roadmap (PRM) planner (Kavraki et al., 1996) is a popular and widely-used path planning algorithm which is a member of sampling based methods. PRM and its variants generate samples to construct an undirected graph over the configuration space where each vertex represents a configuration of the robot without any collision (a member of C_{free}) and each edge demonstrates a collision-free path between configurations. This graph can be thought of as a representation of the free space which is called *probabilistic roadmap*.

During the first phase of PRM (preprocessing), planner samples the configuration space; corresponding to each sample located in C_{free}, PRM algorithm creates a vertex in the roadmap. For each vertex such as q, set $N(q)$ includes the vertexes in the vicinity of q such that the distance between vertexes in $N(q)$ and q is less than a threshold. Then, for each pair of neighbour configurations if the straight line between them is collision free, the edge between them is added to the roadmap graph. There are fast methods to check whether a path between two configurations is collision free or not, for example to check that a connection line is collision free, the points on the line with equal distances, specified by parameter called *resolution*, are checked to be in the free space (Geraerts et al., 2004).

The second phase of PRM is query phase where the roadmap is used to solve path planning queries. The PRM planners are usually multi-query planners and the same roadmap can be used to solve many queries. The result of each query (if exists) is a path between the initial and goal configurations. First, the two configurations are added to the roadmap. Then, the shortest path between them in the graph is returned as the result of the corresponding query. The query phase fails if connecting initial and final configurations to the roadmap fails or if these configurations do not belong to the same connected component in the roadmap. The actual work is done during the preprocessing phase when the roadmap is constructed, which turns out to be the most time-consuming part of the PRM planners.

In a configuration space with obstacles, the shortest path between initial and final points usually traverses the edges of the obstacles. Therefore, the roadmap in the configuration space with obstacles should deem of the points near the edges of the obstacles to increase the ability of PRM to find a near optimal path. For this reason, random sampling exhibit poor efficiency in configurations spaces with obstacles. Consequently, incorporating methods that augment the chance of producing points near the obstacles are of interest. One of these methods is *Gaussian* sampling (Boor et al., 1999) which acts quite well in generating random points near the obstacles. This technique aims at generating two samples near each other such that one is located is the free space while the other one is in $C \setminus C_{free}$.

Most of the PRM planner variations work effectively in simple environments (Amato et al., 1998 Hsu et al., 2003). Too often though, their performances degrade when the environment contains elaborate areas like narrow passages between the obstacles. Narrow passages have relatively small volume comparing to other regions of configuration space; the coverage of such positions with random points is a problem in the PRM approach. Several roadmap construction methods have been proposed to deal with this issue. The most notable ones are filtering and retraction methods (Denny et al., 2014, Lien et al., 2003, Wilmarth et al., 1999). One of the filtering methods to increase the density of random points in narrow passages is

Randomized Bridge Builder (RBB) (Hsu et al., 2003 Sun et al., 2005) sampling, which is developed on top of a test called *Bridge Test*. RBB is considered as an efficient algorithm for finding narrow passages only with a geometric operation.

To reach a more effective roadmap, one needs to do adaptive sampling. In other words, sampling in different regions of configuration space should be done concerning properties of regions and their influence on the optimal path. To this aim, first we propose *configuration space clustering* which randomly splits the configuration space into regions (which may overlap). Then, these regions are clustered based on their features. In other words, different regions are evaluated according to a set of properties and those which exhibit similar properties fall into a same cluster. Later, we propose *Configuration Space Clustering based Sampling* (CSCS) algorithm as an enhancement of hybrid method which adaptively samples each region of configuration space according to the corresponding cluster. Our method enhances the hybrid method (Sun et al., 2005) (one of the popular and basic PRM methods) by finding complex regions of workspace and treating each region according to its complexity. Complex and more critical regions are investigated more thoroughly.

As a proof of concept, CSCS is applied to different simulated environments. Experimental results indicate that CSCS needs less time to build the road map and leads to roadmaps with fewer vertexes which both result in more efficient results in comparison with hybrid method. As a consequence, it seems the proposed clustering method can be used to improve other existing sampling methods such as RRT and its variants (LaValle et al., 1998 Karaman et al., 2010).

This paper is organized as follows. In Section 2 the concept of randomized configuration space clustering and its goals are described. The CSCS algorithm is presented in Section 3. Experimental results are presented and discussed in Section 4. Conclusion and future works are given in Section 5.

2 Randomized Configuration Space Clustering

An efficient algorithm in PRM has following three properties: 1. the output of the algorithm is close to the optimal path. 2. Time complexity of algorithm is reasonable. 3. It is easy to implement. To reach the two first goals one should do adaptive sampling. What is meant by adaptive sampling is that all regions of configuration space are not equally important. Those which are probable to have narrow passages or edges are more noteworthy. As a consequence, sampling in these parts must be fine.

Using adaptive sampling, the roadmap will have a small size which can help routing algorithms to reach a shorter paths as well as minimizing the run time of the planner. Plus, it increases the probability of finding a feasible path from initial point to final point.

Non-adaptive sampling methods such as hybrid algorithm (Sun et al., 2005) have the following problems:

- Consider the entire regions of configuration space equally-important whereas concentrating on complex regions can lead to more efficient results.
- Needs too many random points to cover narrow passages and edges of obstacles.
- Unable to distribute random points uniformly through narrow passages and edges.
- Insufficient distribution of random points around the entrance and exit of narrow passages.
- Unable to find any path in complex environments in efficient time.
- Have a same behaviour for workspaces with different complexity and properties.

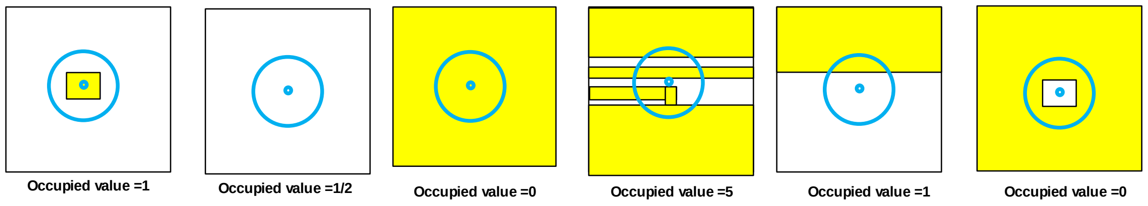

Fig. 1: OV of different circle regions.

- Running bridge test and Gaussian test for all generated samples. In fact all samples are treated equally which is computationally inefficient because those which have small probability of lying on the narrow passages do not need bridge and Gaussian tests while others do.

To reach adaptive sampling, characteristics of all regions of configuration space should be realized and then sampling must be performed according to these characteristics. For this reason, some regions are selected randomly from the configuration space and characteristic and complexity of each region is evaluated through three steps.

First, γ number of circles (sphere and hyper-sphere in higher dimensions) are generated in configuration space with randomly-selected centres. Second, status of each center and its boundary are explored to roughly approximate characteristics and complexity of the region inside the circle. Third, an *occupied value* (denoted by OV) is assigned to each circle shaped region according to its characteristic. This value determines the approximate complexity of that region. That is, regions with higher OV have higher likelihood of containing narrow passages and edges.

OV for each circle regions is computed as follows:

1. If the entire boundary of region is in C_{free} and the center is on occupied space then OV is equal to 1(Fig 1.a).
2. If the entire boundary and the center of a region are in C_{free} then OV is equal to $\frac{1}{2}$ (Fig 1.b).
3. If whole of boundary of region is in occupied space then OV is equal to zero (Fig 1.c and 1.d).
4. Otherwise the occupied value is equal to the number of continues part of boundary in occupied space (Fig. 1.e and 1.f).

For the sake of simplicity in implementation as well as reducing time complexity, it is essential use simple geometric operations. *Clearance* is of these operations which determine the status of a configuration. If it is a member of C_{free} Clearance function returns zero otherwise it returns one. Computing OV requires identifying status of boundaries which seems to need a more complex geometric operation. To avoid that, the boundary of each region is discretized with equal distances (Fig. 2.b), δ, and the clearance operation is used to identify the status of boundaries.

3 Sampling by Configuration Space Clustering

In previous section by using configuration space clustering concept it was shown how to cluster different regions of configuration space based on their OVs. If we manage to control the quantity of samples in each region based on the OV, we have taken an effective step toward adaptive sampling.

3.1 Algorithm

Here, we propose sampling algorithm called *Configuration Space Clustering based Sampling* (CSCS) which tries to consider different samplings in each region of configuration space based on the complexity of the corresponding region. The pseudo code of this algorithm is shown in algorithm 1. First CSCS randomly chooses a number of circles with fixed radiuses in the configuration space. Then it samples in each region based on the OV (γ denotes the number of circles and L denotes the length of radius).

Algorithm 1 CSCS

```
1:  for $i = 1 \to \gamma$ do
2:      Pick a point $(X,Y)$ from $C$ uniformly as the center of region
3:      $B = Boundary(X,Y,\delta)$
4:      $OV = Occupied\_value(B,X,Y)$
5:      $T = f(OV)$
6:      if $OV \neq 0$ & $OV \neq \frac{1}{2}$ then
7:          for $t = 1 \to T$ do
8:              Generate a random point $P$, in the circle with the center $(X,Y)$ and radious $L$
9:              if $CLEARANCE(P)$ is true then
10:                 add $P$ to roadmap set
11:             end if
12:             $RBB(P)$ and $Gaussian(P)$
13:         end for
14:     end if
15:     if $OV \neq \frac{1}{2}$ then
16:         for $t = 1 \to T$ do
17:             A random point $P$, generated on the boundary with the center located at $(X,Y)$
18:             $RBB(P)$ and $Gaussian(P)$
19:         end for
20:     end if
21: end for
```

During each of CSCS iterations, a point is randomly chosen in the configuration space as the center of a circular region (line 2). To compute OV of the corresponding region, first its boundary is discretized (line 3), δ describes the distance between the point in discretizing process of CSCS, then function *occupied_value* computes the value of OV (according to procedure proposed in the previous section) and T number of samples are chosen in the corresponding region (lines 7 to 22). Moreover, *occupied_value* adds a set of configurations to the roadmap (Fig. 2.c). Value of T is assigned based on f(OV) (line 5) which can be a lookup table. If sampled configurations are members of C_{free} they are added to the roadmap (line 10). Clearance function is use to determine whether a configuration is a member of C_{free} or not. Gaussian and Bridge tests are applied to free configurations to add new samples to the roadmap (line 12).

If OV value of a region is equal to $\frac{1}{2}$ then some random samples are generated on the boundary line and both tests are applied as well (lines 15 to 20). Fig. 2 illustrates a schematic of some operations of CSCS.

3.2 Parameter Assignment

CSCS uses three parameters L, δ and γ as well as the function f which must be determined before applying the algorithm. Increase in γ, δ and function f beside decrease in lead to more efficient results but also lead to growth in the roadmap size which results in more computations. Each of these parameters has a special

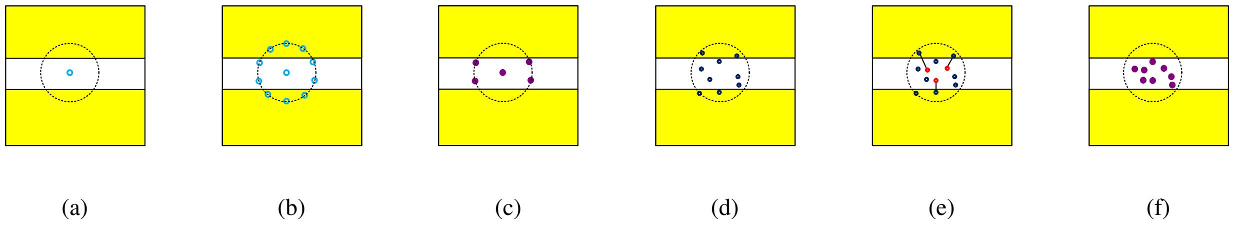

Fig. 2: Schematic of CSCS operations. In (c) and (f) a subset of configurations are added to the roadmap set.

meaning and influence on the algorithm. Therefore, if some information about the configuration space is available beforehand, it is feasible to find a good setting for these parameters. Otherwise, one approach is to use different settings of parameters and find the one that fits the best to the problem (similar to meta-heuristic methods).

3.3 Properties

Table 1 shows how CSCS algorithm with adaptive sampling deals with some deficiency of non-adaptive methods. In a nutshell, CSCS does not generate samples blindly, and consider each region complexity and acts accordingly.

3.4 High Dimensions

CSCS exhibits variable behaviour over the configuration space. This is a desirable property specially when the environment is complex or we are facing with high dimensional problems. Besides, CSCS is easy to implement since it incorporate simple structure and instructions. Therefore, it seems that using CSCS for high dimensional problem is a straightforward task. But as it turns out, computing OVs in problems with more than 4 dimensions is usually intractable which may turn CSCS into an inefficient method in these set of problems.

4 Experiment and Results

This section is devoted to experimental study of CSCS in comparison with hybrid algorithm (Sun et al., 2005). Both methods were studied in same environments as well as equal resolutions and thresholds. Each method was executed 30 times in each of the 3 different environments which are shown in Fig. 3.

Efficiency and duration of each algorithm is dependent to the number of Clearance calls, Bridge and Gaussian (B & G) functions calls as well as number of initial random points and roadmap size. As a result, to gauge the performance of both algorithms, we measured each of these parameters over different executions. Means and variances of these parameters are shown in Table 2. To have a statistical comparison, ANOVA has been used. *P*-values (PV) of statistical tests are also shown in Table 2.

According to Table 2, in 2nd and 3rd configuration spaces, CSCS has reached more efficient results in terms of length with fewer operations (number of initial random points, number of Bridge, Gaussian and Clearance calls). But it has not made any significant improvement in the first environment. Thus, as environments gets more complex, CSCS performance boosts.

Another measure of performance in PRM algorithms is the probability of a response for a query which is sometimes referred to as probabilistic completeness. Here, both algorithms were run in 3 environments shown in Fig. 3 and the percentages of queries with a result are shown in Table 3.

Table 1: Checks existence of edges and narrow passages only in regions which have more complexity.

Problem	How CSCS solves it
Considering almost all regions of configuration space similarly without noting the complexity of parts and their effects on the optimal path.	Using random clustering in configuration space and sampling based on clusters properties.
Needs many random points to cover narrow passages and edges of obstacles.	Producing more random points in complex regions in comparison with other areas as a result of assigning high values of $f(OV)$ to such regions.
Unable to distribute random points uniformly though narrow passages and edges. Insufficient distribution of random points around the entrance and exit of narrow passages. Unable to find any path in complex environments in efficient time	Points are distributed in each part separately and numbers of points in important regions are very high because of high values of $f(OV)$. Therefore, it is more reliable for complete and uniform coverage of narrow passages and edges.
Running bridge and Gaussian tests for all generated samples. In fact all samples are treated equally which is computationally inefficient because those which have small probability of lying along the narrow passages do not need bridge and Gaussian tests while others do.	Checks existence of edges and narrow passages only in regions which have more complexity.

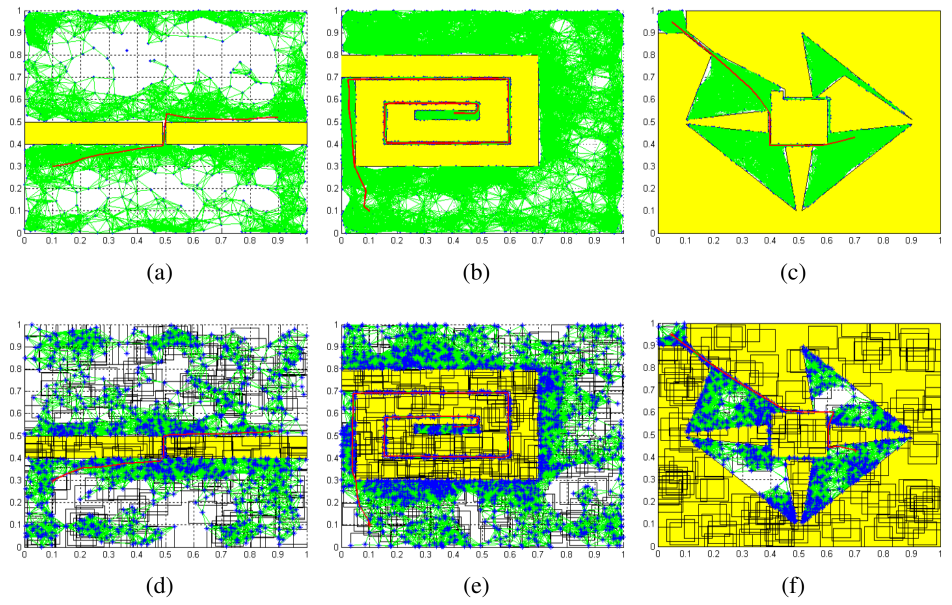

Fig. 3: Instances of roadmap graphs constructed with Hybrid (top row) and CSCS (bottom row) with corresponding outputs in three different environments.

Table 2: Comparison between the outputs returned by CSCS and hybrid algorithm in three configuration spaces (CS)

CS	Method	Path Optimality			Efficiency											
		path length			CLEARANCE calls			roadmap size			B & G calls			initial random point		
		Mean	Std	PV	Mean	Std	PV	Mean	Std	PV	Mean	Std	PV	Mean	Std	PV
1	**hybrid**	0.9524	0.011	3.2	536164	40659.2	1.67	976.76	135.01	0.63	720	0	9.99-	7200	0	0
	CSCS	0.9349	0.040	4	675503	109231	-e8	963.86	60.92		1756	518.94	e16	865.6	55.9	
2	**hybrid**	2.4930	0.017	0.1	2468399	931424	0.00	2212.9	38.23	0	5102	0	0.21	11500	0	0
	CSCS	2.4899	0.021	7	1730259	366892		1687.63	88.02		4419	2997.5		2028	95.6	
3	**hybrid**	0.9961	0.018	0.0	1176409	86901.5	0.37	1039.8	104.99	0.18	4824	0	0.03	6600	0	0
	CSCS	0.9757	0.026	0	1198822	1183356		1069.93	68.67		4021	2004.2		1223	79.6	

Table 3: Success percentage of finding a feasible path by hybrid and CSCS algorithm

Configuration Space	CSCS	Hybrid
1	%90	%70
2	%96.66	%86.66
3	%93.33	%90

5 Conclusion and Future Works

In this paper, we proposed clustering concept in PRM method. This concept has been introduced as a tool toward adaptive sampling and can be combined with there existing sampling algorithms which may lead to improvement in results. Experiments on proposed adaptive sampling method with clustering, CSCS, show effectiveness of the algorithm both in terms of time efficiency and path optimality (especially in complex environments) which is a result of using adaptive sampling.

Our interest is in developing combination of other existing sampling methods (Karaman et al., 2010) with CSCS or using the proposed clustering method to enhance the results which may lead to decrease in run time and variance of outputs. However, CSCS suffers from parameter dependency. Using learning and heuristic methods to solve this problem is in our future research list. Moreover, as mentioned before, calculating OV for high dimensional problems is still challenging and among our current research.

References

Amato, N. M., Bayazit, O. B. Dale, L. K. 1998, ‘Obprm: An obstacle-based prm for 3d workspaces’.

Boor, V., Overmars, M. H. van der Stappen, A. F. 1999, ‘The gaussian sampling strategy for probabilistic roadmap planners’, *International Conference on Robotics and Automation*, 1018–1023.

Canny, J. 1988, *The complexity of robot motion planning*, MIT press.

Choset, H. M. 2005, *Principles of robot motion: theory, algorithms, and implementation*, MIT press.

Denny, J., Sandstrom, R., Julian, N. Amato, N. M. 2014, ‘A region-based strategy for collaborative roadmap construction’, *Eleventh Workshop on the Algorithmic Foundations of Robotics*.

Geraerts, R. Overmars, M. H. 2004, ‘Sampling techniques for probabilistic roadmap planners’, *Proceedings International Conference on Intelligent Autonomous Systems*.

Hsu, D., Jiang, T., Reif, J. Sun, Z. 2003, ‘The bridge test for sampling narrow passages with probabilistic roadmap planners’, *International Conference on Robotics and Automation* **3**, 4420–4426.

Karaman, S. Frazzoli, E. 2010, ‘Incremental sampling-based algorithms for optimal motion planning’, *arXiv preprint arXiv:1005.0416*.

Kavraki, L. E., Svestka, P., Latombe, J.-C. Overmars, M. H. 1996, ‘Probabilistic roadmaps for path planning in high-dimensional configuration spaces’, *IEEE Transactions on Robotics and Automation* **12**(4), 566–580.

Latombe, J.-C. 1991, *Robot Motion Planning*, Kluwer Academic Publishers, Norwell, MA, USA.

LaValle, S. M. 1998, ‘Rapidly-exploring random trees a new tool for path planning’.

Lien, J.-M., Thomas, S. L. Amato, N. M. 2003, ‘A general framework for sampling on the medial axis of the free space’, *International Conference on Robotics and Automation* **3**, 4439–4444.

Sun, Z., Hsu, D., Jiang, T., Kurniawati, H. Reif, J. H. 2005, ‘Narrow passage sampling for probabilistic roadmap planning’, *IEEE Transactions on Robotics* **21**(6), 1105–1115.

Wilmarth, S. A., Amato, N. M. Stiller, P. F. 1999, ‘Maprm: A probabilistic roadmap planner with sampling on the medial axis of the free space’, *International Conference on Robotics and Automation* **2**, 1024–1031.