# A Random Sampling Scheme for Path Planning

Jérôme Barraquand* Lydia Kavraki† Jean-Claude Latombe† Tsai-Yen Li‡
Rajeev Motwani† Prabhakar Raghavan§

## Abstract

*Several randomized path planners have been proposed during the last few years. Their attractiveness stems from their applicability to virtually any type of robots, and their empirically observed success. In this paper we attempt to present a unifying view of these planners and to theoretically explain their success. First, we introduce a general planning scheme that consists of randomly sampling the robot's configuration space. We then describe two previously developed planners as instances of planners based on this scheme, but applying very different sampling strategies. These planners are probabilistically complete: if a path exists, they will find one with high probability, if we let them run long enough. Next, for one of the planners, we analyze the relation between the probability of failure and the running time. Under assumptions characterizing the "goodness" of the robot's free space, we show that the running time only grows as the absolute value of the logarithm of the probability of failure that we are willing to tolerate. We also show that it increases at a reasonable rate as the space goodness degrades. In the last section we suggest directions for future research.*

## 1 Introduction

Robot path planning has been proven a hard problem [40]. There is strong evidence that its solution requires exponential time in the number of dimensions of the configuration space, i.e., the number of degrees of freedom (dofs) of the robot. This result is remarkably stable: it still holds for rather specific robots, e.g., planar linkages consisting of links serially connected by revolute joints [17] and sets of rectangles executing axis-parallel translations in a rectangular workspace [13, 14]. Though general and complete algorithms have been proposed [6, 42], their high complexity precludes any useful application. This negative result has led some researchers to seek heuristic algorithms. While several such planners solve difficult problems, they also often fail or take prohibitive time on seemingly simpler ones. The fact that their behavior is not well characterized is a major drawback: they cannot be used as blackboxes in larger robot control systems.

The number of dofs beyond which existing complete algorithms become practically useless is low, somewhere between 3 and 5. This means that they cannot be used to compute paths for rigid objects translating and rotating in three dimensions, nor for six-dof manipulator arms, two important cases in practice. On the other hand, robot applications tend to involve more degrees of freedom than ever before. For example, an increasing number of manufacturing workcells use several cooperating robots to augment throughput and flexibility. Cells with more than twenty dofs are no longer exceptions. As costs and time for designing and deploying them become more critical, path planners integrated with CAD systems will be in higher demand to facilitate robot programming. Eventually, planners will run online to allow for non-deterministic sequences of goals and events [31]. Robots in domains other than manufacturing (e.g., medical surgery, space exploration) will also require efficient and reliable path planners. Some non-robotics domains raise a similar need as well. In computer graphics, animation of synthetic actors to produce digital movies or video games requires dealing with several dozen dofs. Here, motion planning may drastically reduce the work of human animators

*Salomon Brothers Int. Ltd., Victoria Plaza, 111 Buckingham Palace Road, London SW1W 0SB, UK. Email: jerome.barraquand@sbil.co.uk.

†Department of Computer Science, Stanford University, Stanford, CA 94305, USA. Email: {kavraki, latombe, motwani}@cs.stanford.edu.

‡Department of Computer Science, National Chengchi University, Wenshan, Taipei, Taiwan. Email: li@cs.nccu.edu.tw.

§IBM Almaden Research Center, San Jose, CA 95120, USA. Email: pragh@almaden.ibm.com.

who currently input large numbers of key frames. In molecular biology, motion planning can help compute plausible docking motions of molecules modeled as spatial linkages with many dofs.

Collision-free path planning, which assumes perfect knowledge of the world and stationary obstacles, is only the most basic motion planning problem in robotics. Clearly, we would ultimately like robot planners to also deal with issues such as uncertainties, moving obstacles, movable objects, and dynamic constraints [29, 30]. But every extension of the basic problem adds in computational complexity. For instance, allowing moving obstacles makes the problem exponential in the number of moving obstacles [6, 41]; uncertainties in control and sensing make the problem exponential in the complexity of the robot environment [6]. Before we can effectively investigate such extensions in large configuration spaces, it seems that we must better understand how to practically solve basic path planning.

Path-planning applications are so diverse that it is infeasible to design a tailor-made algorithm for every possible robot.[1] Instead, we need general path planning algorithms not bound to the specifics of any particular robot. We believe that between the two extreme types of planners suggested above – complete and heuristic – there is place for practically efficient general planners achieving a weaker form of completeness. In other words, we may perhaps trade a limited amount of completeness against a major gain in computing efficiency. Full completeness requires the planner to always answer a path-planning query correctly, in asymptotically bounded time. A weaker, but still interesting form of completeness is the following: if a solution path exists, the planner will find one in bounded time, with high probability. We call it *probabilistic completeness.* This weaker completeness becomes particularly interesting if we can show that the planner's running time grows slowly with the inverse of the failure probability that we are willing to tolerate.

With this philosophy in mind, we have designed new path planners and experimented with them in large configuration spaces. One of them, described in [3, 4, 29], is a potential-field-based planner that escapes local minima by performing random walks; in the following, we will refer to it as the *potential-field planner.* Another planner, presented in [18, 20, 21, 24], precomputes a "roadmap" (network) of simple paths connecting randomly selected configurations and tries to construct a path between any two input configurations by connecting them to this roadmap; we will refer to this planner as the *roadmap planner.* Both these planners have been successfully applied to complex problems. For example, in [26], the potential-field planner was used to automatically synthesize a video clip with graphically simulated human and robot characters entailing a 78-dimension configuration space. Both the potential-field and the roadmap planners have been used to check that parts can be removed from an aircraft engine for inspection and maintenance [8]; here, paths are generated in configuration spaces having only six dimensions, but the parts have particularly complex geometry.

These two planners achieve probabilistic completeness. For the potential-field planner, this property remains qualitative: if there exists a path, the probability that the planner finds one tends toward one as the running time increases; but the convergence speed is unknown. But, for the roadmap planner, we have proven stronger results that relate the probability that it finds a path, when one exists, to its running time [18, 19, 22, 23]. In turn, these theoretical results suggest improvements of the planner.

Other work investigating similar or related randomized planning approaches include [1, 5, 15, 16, 35, 36]. Formal attempts to predict the behavior of specific random planners are proposed in very few papers [9, 28].

This paper proposes a consistent framework to describe and study the randomized planners cited above, with the goal to eventually build more powerful planners. In the planners cited above, the robot's free space is not explicitly represented, but randomly sampled. We believe that this is the central concept underlying their success. In Section 2 we capture this concept into a general computational scheme for path planning in large configuration spaces.[2] In Section 3 we make our discussion more precise by presenting our potential-field and roadmap planners as two instances of planners using this scheme, but applying two different sampling strategies. In Section 4 we give two formal analyses of the probabilistic completeness of variants of the roadmap planner. These analyses provide a theoretical explanation for the empirically observed success of the roadmap plan-

[1] In any case, very few tailor-made planners have been successfully designed for specific robots with more than four dofs.

[2] This scheme can also be applied to configuration spaces having few dimensions; but it is less interesting in that case, since complete algorithms are then available.

ner. They are the crux of this paper. Indeed, as it has become relatively easy to design new randomized path planners that outperform previous ones on some well-selected problems, it is increasingly important to formally explain their strengths and weaknesses. This paper does not present experimental results obtained with our planners. Many such results were previously reported in [3, 8, 18, 20, 21, 24, 26, 27] and in papers by other authors [7, 11, 47]. The reader is referred to [34] for a textbook on randomized algorithms and sampling techniques.

## 2 General Scheme

We consider the problem of planning collision-free paths for an arbitrary $n$-degree-of-freedom holonomic[3] robot $\mathcal{A}$. We let $\mathcal{C}$ denote the configuration space of $\mathcal{A}$ and $\mathcal{C}_{free}$ stand for the open subset of the collision-free configurations in $\mathcal{C}$. We also refer to $\mathcal{C}_{free}$ as the *free space* and to configurations in $\mathcal{C}_{free}$ as *free configurations*. A planning problem is specified by two free configurations, $\boldsymbol{q}_{init}$ and $\boldsymbol{q}_{goal}$, called the *initial* and the *goal* configurations, respectively. Any path lying in $\mathcal{C}_{free}$ that joins these two configurations is a solution of this problem.

We assume for simplicity that $\mathcal{C}$ is the cube $[0,1]^n$, so that each configuration $\boldsymbol{q}$ is described by an $n$-tuple $(q_1,\ldots,q_n)$. However, it is straightforward to extend our presentation to cases where $\mathcal{C}$ is multiply connected, i.e., one or more dimensions can "wrap around". The only requirement is that $\mathcal{C}$ be measurable (in the Lebesgue sense), which is always achieved in practice.

Although there exist algorithms to construct an explicit representation of $\mathcal{C}_{free}$ given the geometry of $\mathcal{A}$ and the obstacles in semi-algebraic form, their time complexity makes them impractical for any sufficiently large value of $n$. On the other hand, reasonably efficient algorithms are available to compute the Euclidean distance between two objects in three-dimensional space (e.g., [10, 32, 37, 39, 45]). This leads us to assume that $\mathcal{C}_{free}$ is implicitly given by a function, CLEARANCE: $\mathcal{C} \rightarrow \mathbb{R}$, that maps any configuration $\boldsymbol{q}$ to the distance (in the workspace) between the robot placed according to $\boldsymbol{q}$ and the obstacles, or between two bodies of the robot, whichever is smaller. In the case where the robot collides with an obstacle or with itself, CLEARANCE$(\boldsymbol{q})$ returns 0 or a negative number. Thus, whenever CLEARANCE$(\boldsymbol{q})$ is positive, $\boldsymbol{q}$ belongs to $\mathcal{C}_{free}$. We refer to CLEARANCE$(\boldsymbol{q})$ as the *clearance* of $\boldsymbol{q}$.

For any bounded robot $\mathcal{A}$, there exists a positive constant $\rho$ such that when $\mathcal{A}$ moves along a straight path in $\mathcal{C}$ between any two configurations $\boldsymbol{q}=(q_1,\ldots,q_n)$ and $\boldsymbol{q}'=(q'_1,\ldots,q'_n)$, no point of $\mathcal{A}$ traces a curve segment longer than $\rho\max_{i\in[1,n]}|q_i-q'_i|$. We assume here that $\rho$ is given, though for most usual robots its computation is straightforward and can be included in the planner. Rather than using a single $\rho$, we may also partition $\mathcal{C}$ into several regions and have a distinct constant $\rho_i$ for each region. If the $\rho_i$'s differ by large amounts, this could substantially reduce the running time of a planner based on the scheme presented below, but we will not go into such detail in this paper.

**Definition 1** *Let $\boldsymbol{q}=(q_1,\ldots,q_n)$ and $\boldsymbol{q}'=(q'_1,\ldots,q'_n)$ be two free configurations whose respective clearances are $\eta$ and $\eta'$. They are said to be adjacent if $\rho\max_{i\in[1,n]}|q_i-q'_i|<\max\{\eta,\eta'\}/2$.*

Clearly, if two configurations are adjacent the straight-line segment joining them lies entirely in $\mathcal{C}_{free}$. In fact, the coefficient 1/2 appearing on the right-hand side of the above inequality is intended to make sure that the robot cannot collide with itself, since two bodies might then move simultaneously toward each other. This coefficient can be removed if self-collisions are known in advance to be impossible. Note also that the map:

$$(\boldsymbol{q},\boldsymbol{q}')\in\mathcal{C}^2\mapsto\max_{i\in[1,n]}|q_i-q'_i|\in\mathbb{R}^+\cup\{0\}$$

is the $L^\infty$ distance in $\mathcal{C}$. We could choose other distances in $\mathcal{C}$ to define adjacency. Of course, the constant $\rho$ depends on this choice.

Now we can state our **general planning scheme** as follows:

> Pick configurations in $\mathcal{C}$ at random; retain those configurations which lie in $\mathcal{C}_{free}$ (along with $\boldsymbol{q}_{init}$ and $\boldsymbol{q}_{goal}$) as the nodes of a graph $G$; and connect adjacent configurations by links of $G$.
>
> Return YES if $\boldsymbol{q}_{init}$ and $\boldsymbol{q}_{goal}$ belong to the same connected component of $G$.
>
> Return NO if no path has been found after having generated $c$ configurations, where $c$ is an input parameter.

The answer YES is always correct, that is, whenever the planner returns YES, there actually exists a collision-free path connecting $\boldsymbol{q}_{init}$

[3] Our presentation can be extended to nonholonomic robots. See [36, 43, 44].

to $\boldsymbol{q}_{goal}$; moreover such a path can easily be extracted from the graph $G$. But the answer NO is not necessarily correct. Indeed, after a finite amount of computation (defined by the parameter $c$), the planner may still not have found a path between $\boldsymbol{q}_{init}$ and $\boldsymbol{q}_{goal}$, even if one exists.

The key component of this planning scheme is the *sampling strategy* applied to generate the nodes of $G$. Different strategies are possible. For example, if multiple planning queries are to be made with the same robot and obstacles (*multi-query case*), then it may be suitable to invest preprocessing time to construct a network of configurations (which we call a *roadmap*). Processing each query then only requires connecting the input initial and goal configurations to the roadmap. If the queries are not known in advance, it seems reasonable to construct the roadmap by choosing configurations uniformly at random from $[0,1]^n$ since the resulting free configurations are then uniformly distributed over $\mathcal{C}_{free}$. However, we will see in Section 3.2 that, while the roadmap is being generated, it is possible to derive heuristic information from it and bias the selection of the new configurations.

Instead, if a single or very few queries are to be made with the same robot and obstacles (*single-query case*), the planner may be more successful by using a sampling strategy that picks new configurations, first in neighborhoods of the initial and goal configurations and then, iteratively, in neighborhoods of the newly generated configurations, until the two "sampling waves" meet. A similar strategy could also be used in the multi-query case to connect the initial and goal configurations to the roadmap.

Although the efficiency of any planner may be evaluated through experimentation, formal analysis is desirable to compare planners and stress their strengths and weaknesses. Ideally a planner's outcome should be YES with the highest probability whenever a free path exists and this outcome should be generated in minimal time. Hence, the analysis should relate the probability that the planner produces an incorrect answer to its running time. Given a small positive number $\alpha$ (the acceptable probability that the planner's output is NO while a free path exists), we would like to bound the running time of the planner by a function of $\alpha$. If this function grows slowly with $\alpha^{-1}$, then the planner is particularly interesting since we can get arbitrarily close to full completeness at a reasonable cost.

Let us assume for simplificity that CLEARANCE takes constant time to evaluate. Then the planner's running time mainly depends on two numbers: the number of sample configurations and the number of pairs of free configurations checked for adjacency. The number of sample configurations is at most $c$. But only a fraction of them, $f$, are in $\mathcal{C}_{free}$, hence in the constructed graph $G$. In practice, it is often the case that $f \ll c$. The planner should thus strive to get the greatest ratio $f/c$, but the distribution of the generated free configurations is also crucial to eventually find a path. The number of adjacency checks is at most proportional to $f^2$; however, the planner may choose not to test all pairs of free configurations for adjacency.[4]

It seems likely that no strong property can be proven for any given planner, if we do not make some assumption about $\mathcal{C}_{free}$. Moreover, no single planner is likely to be the most efficient for all possible problems. This suggests that planners should be analyzed under some well-specified assumptions. In Section 4, we will study the work carried out by a two-phase planner under two distinct assumptions. In one, the *visibility volume assumption*, $\mathcal{C}_{free}$ is such that every free configuration "sees" a subset of $\mathcal{C}_{free}$ whose volume is at least an $\epsilon$ fraction of the total volume of $\mathcal{C}_{free}$ (we then say that $\mathcal{C}_{free}$ is $\epsilon$-good). In the second assumption, the *path clearance assumption*, there exists a collision-free path between $\boldsymbol{q}_{init}$ and $\boldsymbol{q}_{goal}$ that has some given clearance $\xi$. Assumptions must be carefully crafted. Indeed, if these are too specific or unrealistic, the analysis will not yield useful results; on the other hand, if they are too general, the results will be too weak.

# 3 Specific Planners

We now present two planners that make use of the above scheme with different sampling strategies.

## 3.1 Potential-Field Planner

This planner was originally described in [3, 4, 29], along with experimental results.[5] Extensions and additional experimental results were presented

[4] Checking the sample configurations for adjacency is an instance of the orthogonal range-searching problem [2, 38]. Using the range-tree technique, the pairs of adjacent configurations can be computed in $O(f \log^{n-1} f)$ time, or slightly faster. This is better than $O(f^2)$ when $f$ is large enough.

[5] The planner is available by anonymous ftp from `flamingo.stanford.edu:/pub/li/rpp3d.tar.gz`.

in [8, 11, 26, 27]. Here, we focus on its relation to the general scheme presented above.

The planner is given a function $\boldsymbol{U} : \mathcal{C}_{free} \rightarrow \mathbb{R}^+ \cup \{0\}$, the *potential field*, with a single global minimum (0) at the goal configuration.[6] It then attempts to connect $\boldsymbol{q}_{init}$ to $\boldsymbol{q}_{goal}$ by alternating *down motions* and *escape motions*. Each down motion "descends" along $\boldsymbol{U}$ until it reaches a local minimum, while escape motions attempt to flee from local minima.

The following algorithm, in which $h$ is an input parameter, constructs a down motion starting at configuration $\boldsymbol{q}_s$:

`Down-Motion`($\boldsymbol{q}_s$):

1. $\boldsymbol{q} \leftarrow \boldsymbol{q}_s$.
2. Until $\boldsymbol{q}$ is not labelled as a local minimum do:
   (a) Pick at random up to $h$ configurations adjacent to $\boldsymbol{q}$, until one of them, $\boldsymbol{q}'$, satisfies $\boldsymbol{U}(\boldsymbol{q}') < \boldsymbol{U}(\boldsymbol{q})$.
   (b) If the previous step succeeded in generating $\boldsymbol{q}'$, then reset $\boldsymbol{q}$ to $\boldsymbol{q}'$; else label $\boldsymbol{q}$ as a local minimum.
3. Return $\boldsymbol{q}$.

At Step 2(a) let $\boldsymbol{q} = (q_1, \ldots, q_n)$ and $\eta =$ CLEARANCE($\boldsymbol{q}$). Configurations are picked at random from the volume defined by $\Pi_{i=1,\ldots,n}[q_i - \eta/\rho, q_i + \eta/\rho] \cap [0,1]^n$. According to Definition 1, all configurations in this volume are adjacent to $\boldsymbol{q}$. To guarantee that Step 2 does not loop for ever, $\boldsymbol{U}$ must have no local minimum in the boundary of $\mathcal{C}_{free}$. This can easily be obtained by including in the definition of $\boldsymbol{U}(\boldsymbol{q})$ a term proportional to 1/CLEARANCE($\boldsymbol{q}$).

The following algorithm constructs an escape motion starting at a local minimum $\boldsymbol{q}_l$:

`Escape-Motion`($\boldsymbol{q}_l$):

1. Pick at random the length, $\ell$, of the motion.[7]
2. $\boldsymbol{q} \leftarrow \boldsymbol{q}_l$; $l \leftarrow 0$.
3. While $l < \ell$ do:
   (a) Pick at random a free configuration $\boldsymbol{q}'$ adjacent to $\boldsymbol{q}$.
   (b) $l \leftarrow l + \max_i |q_i - q_i'|$.
   (c) $\boldsymbol{q} \leftarrow \boldsymbol{q}'$.
4. Return $\boldsymbol{q}$.

The overall planning algorithm is the following:

`Potential-Field-Planner`($\boldsymbol{q}_{init}, \boldsymbol{q}_{goal}$):

1. $\boldsymbol{q}_l \leftarrow$ `Down-Motion`($\boldsymbol{q}_{init}$).
2. While $\boldsymbol{q}_l \neq \boldsymbol{q}_{goal}$ do:
   (a) Do:
      i. If the total number of configurations generated so far exceeds $c$ then return NO and halt.
      ii. $\boldsymbol{q}_s \leftarrow$ `Escape-Motion`($\boldsymbol{q}_l$);
      iii. $\boldsymbol{q}_l' \leftarrow$ `Down-Motion`($\boldsymbol{q}_s$);

      until $\boldsymbol{U}(\boldsymbol{q}_l') < \boldsymbol{U}(\boldsymbol{q}_l)$.
   (b) $\boldsymbol{q}_l \leftarrow \boldsymbol{q}_l'$.
3. Return YES.

The graph $G$ constructed by the planner is a tree of paths rooted at $\boldsymbol{q}_{init}$. Each configuration is generated such that it is adjacent to a previously generated configuration, and is only connected to this configuration. Hence, the planner does not test pairs of configurations for adjacency, avoiding the cost of this test. During a down motion, the potential $\boldsymbol{U}$ introduces a bias in the choice of the tree path that will be followed by the planner. Although the global geometry of $\mathcal{C}_{free}$ is unknown, $\boldsymbol{U}$ can be seen as a *specialist* that gives some heuristic indications about this geometry, i.e.: which directions are promising and which aren't. The techniques in [3, 29] compute $\boldsymbol{U}$ by combining local-minima functions computed over the robot's workspace. Although the resulting $\boldsymbol{U}$ often prevents the planner from getting trapped into big obstacle cavities, it still has local minima and therefore is an imperfect characterization of the free space's geometry. On the other hand, computing local-minima-free potentials in large dimensional spaces is a difficult problem [25] that is at least as hard as path planning itself.

Using well-known properties of random motions, the potential-field planner can be shown probabilistically complete [3, 28]. Moreover, a calculation of the finite expected number of invocations of `Escape-Motion` is given in [28]. The idea underlying this calculation is the following. The basins of attraction of the local minima of $\boldsymbol{U}$ which have non-zero measure (the goal basin

[6] Techniques to automatically construct this function are proposed in [3, 29].

[7] In [3], we note that, on the average, a random walk of length $\ell$ ends $\sqrt{\ell}$ away from its starting point. This leads us to suggest choosing $\sqrt{\ell}$ according to a truncated Laplace distribution with mean value equal to the "radius" of the configuration space. Here, this radius is 1.

is assumed to be one of them) form a partition of the free space. For any two such basins $B_i$ and $B_j$, one can define the finite transition probability $p_{ij}$ that a random walk generated by `Escape-Motion` from the minimum in $B_i$ terminates into $B_j$. The expected number of invocations of `Escape-Motion` before entering the goal basin is then expressed as a function of the $p_{ij}$'s. In theory, this result gives some indication of the expected complexity of the potential-field planner. But, in practice, the $p_{ij}$'s are unknown. In fact, a key issue in analyzing a planner based on random sampling is to translate some geometric property of the free space into probabilistic terms.

The potential-field planner has successfully solved many difficult problems (e.g., see [3, 8, 11, 26]). However, it is easy to create problems that it fails to solve in a reasonable amount of time, though they admit rather obvious solutions. Not surprisingly, most failures are due to *traps*, which cause some transition probabilities $p_{ij}$ to be very small (in particular, the probabilities to enter the goal basin from some other basins). A trap is a basin of attraction of a local minimum of $\boldsymbol{U}$ that is almost completely surrounded by forbidden configurations (i.e., $\mathcal{C}\backslash\mathcal{C}_{free}$). Hence, a trap admits only narrow exits and the potential function inside the trap guides the robot away from these exits. If the robot's path starts within a trap or is guided into one by $\boldsymbol{U}$, each escape motion executed then has a tiny probability of escaping the local minimum. One could imagine a variant of the planner where the potential field is randomly guessed among a collection of several functions and changed several times during planning, hoping that the various potentials would entail different traps. We did some experiments with this idea, but we only got limited results.

### 3.2 Roadmap Planner

The problems encountered with the potential-field planner led us to develop another planner, the roadmap planner[8] described in [18, 20, 21, 24]. Several variants of this planner have been implemented; we present one below. Unlike the potential-field planner, this new planner uses no problem-specific heuristics.

The roadmap planner operates in two phases, preprocessing and query processing: The preprocessing phase consists of constructing a network $R$ of configurations, the roadmap. The configurations in $R$ are called *milestones*. They only form a subset of all the sample configurations generated by the planner; hence, the roadmap is not exactly the graph $G$ mentioned in the general scheme. Every query specifies two configurations, $\boldsymbol{q}_{init}$ and $\boldsymbol{q}_{goal}$, in $\mathcal{C}_{free}$. Processing the query consists of connecting these configurations to two milestones and checking that these two milestones are in the same connected component of $R$.

The planner uses a simple algorithm, `Connect` (also called the *connector*), to construct the links of $R$. Given any two milestones, $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$, the connector checks if the straight-line segment connecting them lies in $\mathcal{C}_{free}$. It does so by dichotomically breaking the segment into shorter segments and checking the endpoints of each segment for adjacency. It stops breaking a segment whenever its two endpoints are adjacent or one of them is not in free space. In the second case, `Connect` halts, without generating a link between $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ in the roadmap. Instead, if the connector eventually succeeds in partitioning the segment $\boldsymbol{m}_1\boldsymbol{m}_2$ into segments (possibly, of unequal lengths) such that the endpoints of each segment are adjacent, it establishes a link between $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ in the roadmap. The work done by the connector is part of the sampling work done by the roadmap planner, but the sample configurations it generates are not permanently stored; they are not part of the roadmap $R$. In other words, the planner does not attempt to find additional connections among sample configurations generated between milestones. To keep our presentation simple, we assume that `Connect` only tries straight paths, but it could try other canonical paths as well. We say that a configuration *sees* another configuration if the straight-line segment joining them lies entirely in $\mathcal{C}_{free}$.

In the following algorithm we could limit the total number of sample configurations to $c$, as in the general scheme of Section 2. However, putting the limit on the number of generated milestones instead is more convenient; it will also facilitate the analyses of Section 4. The preprocessing phase constructs an initial roadmap containing $r$ milestones (Steps 2 and 3 in the algorithm shown below). Then it expands this initial roadmap into a final one containing $s > r$ milestones. Both $r$ and $s$ are input parameters. The first $r$ milestones are chosen uniformly over $\mathcal{C}_{free}$. The remaining $s - r$ milestones are selected in small regions considered as "difficult" parts of $\mathcal{C}_{free}$. The preprocessing algorithm is the following:

[8]The planner is available by anonymous ftp from `flamingo.stanford.edu:/pub/kavraki/prm.tar.gz`.

`Preprocessing`:

1. $i \leftarrow 0$.
2. While $i < r$ do:
   (a) Pick a configuration $\boldsymbol{q}$ in $\mathcal{C}$ at random.
   (b) If CLEARANCE($\boldsymbol{q}$) $> 0$ then
      i. Store $\boldsymbol{q}$ as a milestone of the roadmap.
      ii. $i \leftarrow i + 1$.
3. For every pair of milestones $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ whose distance is less than $d$, do: `Connect`($\boldsymbol{m}_1, \boldsymbol{m}_2$).
4. Invoke `Resample` to expand the roadmap by $s - r$ milestones (see below).

The threshold $d$ at Step 3 is an input parameter. It is used to limit the number of pairs of milestones that are tested by `Connect`, since in most spaces two milestones that are far apart are unlikely to see each other. Step 3 takes time at most quadratic in $r$, which is usually much smaller than the total number of sample configurations generated by the roadmap planner.[9]

`Resample` selects milestones with probability inversely proportional to their degrees in the roadmap (i.e., their number of connections to other milestones). Intuitively, a milestone with few or no connections lies in a difficult region of the free space. For each milestone $\boldsymbol{m}$ that it selects, `Resample` picks a number of new milestones at random in a neigborhood of $\boldsymbol{m}$ and invokes `Connect` to try to link each of these new milestones to other milestones. This expansion of the roadmap terminates when the total number of milestones is $s$. Experiments have been done with and without the roadmap expansion step (Step 4). Roadmaps containing the same total number of milestones have been constructed both ways. The roadmaps generated using the expansion step have been consistently better, i.e., subsequent queries were processed more quickly with less incorrect NO answers. Over a large range of problems, generating 2/3 of the milestones at Step 2 and 1/3 at Step 4 gave good results.

After preprocessing, each query is handled as follows ($g$ is an input parameter):

`Query-Processing`($\boldsymbol{q}_{init}, \boldsymbol{q}_{goal}$):

1. For $i = \{init, goal\}$ do:
   (a) If there exists a milestone $\boldsymbol{m}$ that sees $\boldsymbol{q}_i$ then $\boldsymbol{m}_i \leftarrow \boldsymbol{m}$.
   (b) Else
      i. Repeat $g$ times:
         Pick a configuration $\boldsymbol{q}$ at random in the neighborhood of $\boldsymbol{q}_i$
         until $\boldsymbol{q}$ sees both $\boldsymbol{q}_i$ and a milestone $\boldsymbol{m}$.
      ii. If all $g$ trials failed then return NO and halt, else $\boldsymbol{m}_i \leftarrow \boldsymbol{m}$.
2. If $\boldsymbol{m}_{init}$ and $\boldsymbol{m}_{goal}$ are in the same component of the roadmap then return YES; else return NO.

[9] Footnote 4 applies here too.

# 4 Analysis

We now give formal analyses of two variants of the roadmap planner. We characterize the amount of computation that the planner must do in order to give correct answers with high probability.

The intuitive reason for the experimental success of the above planners is that there usually exist many collision-free paths joining two configurations. Hence, to bound the running time of the roadmap planner, we assume that $\mathcal{C}_{free}$ satisfies some geometric property capturing the above intuition. We propose two such properties. Our thesis is that the success of any sampling strategy will stem from a similar property.

In both analyses, we consider that the key number affecting the planner's running time is the number of milestones in the constructed roadmap. Since to generate a single milestone it might be necessary to randomly pick several (possibly, many) configurations in $\mathcal{C}$, we implicitly assume that the volume of $\mathcal{C}_{free}$ relative to the volume of $\mathcal{C}$ is not too small. If this assumption is not satisfied, any variant of the roadmap algorithm presented in Section 3.2 will behave poorly. In this case, we should probably make the additional assumption that a few free configurations are given (after all, every query will give two such configurations); then, a possible sampling strategy could be to build a roadmap by generating milestones in small regions centered at the given configurations, first, and at the newly generated milestones, next.

Note that bounding the number of milestones is not sufficient to bound the running time of the planner, because it does not account for the running time of `Connect`. If two milestones passed to `Connect` see each other, but the line segment connecting them is arbitrarily close to the free space boundary, then the connector will have to break it into arbitrarily many small segments. This problem could be eliminated by extending the general scheme of Section 2 with a primitive computing the volume swept out by the robot when it moves between two configurations along a straight path in configuration space.

Here, we only state the main results of our analyses. We give proofs of three selected theorems (1, 2, and 7) in the appendix. For more details and additional proofs, we refer the reader to [22, 23] (first analysis) and to [18, 19] (second analysis).

In the rest of this section we denote the volume of a subset $\mathcal{X}$ of $\mathcal{C}$ by $\mu(\mathcal{X})$.

## 4.1 Visibility Volume Assumption

For any configuration $\boldsymbol{q} \in \mathcal{C}_{free}$, let $\mathcal{S}(\boldsymbol{q})$ consist of all those configurations $\boldsymbol{q}' \in \mathcal{C}_{free}$ that $\boldsymbol{q}$ sees.

**Definition 2** *Let $\epsilon$ be a positive real. A configuration $\boldsymbol{q} \in \mathcal{C}_{free}$ is $\epsilon$-good if $\mu(\mathcal{S}(\boldsymbol{q})) \geq \epsilon\mu(\mathcal{C}_{free})$. Furthermore, $\mathcal{C}_{free}$ is $\epsilon$-good if all the configurations it contains are $\epsilon$-good.*

The visibility volume assumption made here is that $\mathcal{C}_{free}$ is $\epsilon$-good, that is, each configuration in it sees a significant portion of $\mathcal{C}_{free}$. The underlying intuition is that it is then relatively easy to pick a set of milestones that, collectively, can see most of $\mathcal{C}_{free}$. But the assumption fails to prevent $\mathcal{C}_{free}$ from containing narrow passages through which it might be difficult to connect milestones. For example, consider the case where $\mathcal{C}$ is two-dimensional and $\mathcal{C}_{free}$ consists of two disks of equal size that overlap by a very small amount. Then $\mathcal{C}_{free}$ is $\epsilon$-good for $\epsilon \approx 0.5$. But the probability that any milestone in one disk sees a milestone in the other disk is very small. For this reason, we use a variant of the roadmap algorithm that embeds a "complex planner". We assume that this complex planner is error-free in that it discovers a path between two given configurations whenever one exists, and reports failure when there is none. But of necessity such a complete planner must be expensive to run, and so we seek to use it sparingly. In particular, to keep query processing fast, we restrict the use of the complex planner to the preprocessing stage.

The preprocessing algorithm is similar to the one given in Section 3.2. The main difference is that Step 4 invokes `Permeation`, instead of `Resample`. `Permeation` uses the complex planner to improve the roadmap connectivity; unlike `Resample`, it does not add new milestones to the roadmap. The constructed roadmap contains $s$ milestones; these are all generated at Step 2.

`Preprocessing`:
1. $i \leftarrow 0$.
2. While $i < s$ do:
   (a) Pick a configuration $\boldsymbol{q}$ in $\mathcal{C}$ at random.
   (b) If CLEARANCE($\boldsymbol{q}$) $> 0$ then
      i. Store $\boldsymbol{q}$ as a milestone of the roadmap.
      ii. $i \leftarrow i + 1$.
3. For every pair of milestones $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$, do: `Connect`($\boldsymbol{m}_1, \boldsymbol{m}_2$).
4. Pick one representative milestone from each component of the current roadmap. Let $V$ be the set of these representative milestones. Invoke `Permeation`($V$) to improve the connectivity of the milestones (see below).

The result of this preprocessing is a roadmap such that any two nodes are in the same component if and only if the corresponding milestones are in the same component of $\mathcal{C}_{free}$. Step 3 may fail to find all possible links between the milestones due to the incompleteness of the connector. `Permeation` invoked in Step 4 fixes this problem by using the complex planner to discover additional connections between milestones. Note that `Permeation` is a last resort; hopefully, much if not all of the connectivity information should have been discovered before this step.

It is worth noticing that the algorithms of the implemented planner presented in [18, 20, 21] are similar to those described here, except that it uses the potential-field planner of Section 3.1 instead of the complex planner. The potential-field planner is not complete, but it is quite rare in practice that two milestones in the same component of the free space can be connected neither by invoking `Connect` at Step 3 of the proprocessing, nor by the potential-field planner. The planner of [18, 20, 21] seems a good approximation of the algorithms given here, and the analysis proposed below is relevant to this planner.

The query processing is handled by the following algorithm, which is similar to the query-processing algorithm of Section 3.2:

`Query-Processing`($\boldsymbol{q}_{init}, \boldsymbol{q}_{goal}$):
1. For $i = \{init, goal\}$ do:
   (a) If there exists a milestone $\boldsymbol{m}$ that sees $\boldsymbol{q}_i$ then $\boldsymbol{m}_i \leftarrow \boldsymbol{m}$.

(b) Else

i. Repeat $g$ times:

Pick a configuration $\boldsymbol{q}$ at random in $\mathcal{S}(\boldsymbol{q}_i)$

until $\boldsymbol{q}$ sees both $\boldsymbol{q}_i$ and a milestone $\boldsymbol{m}$.

ii. If all $g$ trials failed then return FAILURE and halt, else $\boldsymbol{m}_i \leftarrow \boldsymbol{m}$.

2. If $\boldsymbol{m}_{init}$ and $\boldsymbol{m}_{goal}$ are in the same component of the roadmap then return YES; else return NO.

Step 1(b)i differs slightly from the corresponding step in the query-processing algorithm of Section 3.2. Since in general we do not know how to compute $\mathcal{S}(\boldsymbol{q}_i)$, picking configurations in this set requires guessing configurations in configuration space and retaining only those that are seen by $\boldsymbol{q}_i$. This means that each of the $g$ iterations performed at Step 1(b)i may require several trials to obtain a configuration visible from $\boldsymbol{q}_i$. The analysis proposed below ignores these trials and focuses only on the number $g$.

Another difference between the above algorithm and the one of Section 3.2 is that it returns FAILURE (instead of NO) at Step 1(b)ii. Due to the use of the `Permeation` in the preprocessing, both the answers YES and NO are now always correct. With some probability though, the query-processing algorithm may fail to give an answer.

Let us first present some performance guarantee for the preprocessing phase. Call a set of milestones *adequate* if the volume of the subset of $\mathcal{C}_{free}$ not visible from any of these milestones is at most $(\epsilon/2)\mu(\mathcal{C}_{free})$. Intuitively, if we were to place a point source of light at each milestone, we would like a fraction at least $1-\epsilon/2$ of $\mathcal{C}_{free}$ to be illuminated.

**Theorem 1** *Let $\beta \in (0,1]$ be a positive real constant. Let $C$ be a positive constant large enough that for any $x \in (0,1]$, $(1-x)^{(C/x)\log(4/x\beta)} \leq x\beta/4$. If $s \geq (C/\epsilon)(\log 1/\epsilon + \log 4/\beta)$, then preprocessing generates an adequate set of milestones with probability at least $1-\beta$.*

Note that as $\epsilon$ increases, the requirement for adequacy grows weaker, i.e., the portion of the free space that has to be visible by at least one milestone gets smaller. Intuitively, this comes from the fact that a greater $\epsilon$ will make it easier to connect query configurations to the roadmap. Naturally, the number of milestones needed becomes smaller.

Theorem 1 only says that most of $\mathcal{C}_{free}$ is likely to be visible from some milestone in the roadmap; using this property alone, we can show that queries can be answered quickly. But the adequacy of the milestones is not sufficient to imply that the roadmap is a good representation of the connectivity of $\mathcal{C}_{free}$. The use of the complex planner in the permeation algorithm is inevitable to ensure a good probability that the query processing outcomes YES or NO.

Let us choose $g = \log(2/\gamma)$ at Step 2 of `Query-Processing`, where $\gamma \in (0,1]$ is the failure probability we are willing to tolerate during a query. We can show the following performance guarantee for the query processing phase:

**Theorem 2** *If the set of milestones chosen during preprocessing is adequate, then the probability that the query-processing algorithm outputs* FAILURE *is at most $\gamma$.*

In fact, our analysis (in the Appendix, Section B) implies that the expected number of executions of Step 1(b)i in the query-processing algorithm is at most 2.

Theorems 1 and 2 give performance guarantees for the preprocessing phase and the query-processing phase, respectively. This is appropriate, since many queries will be made using the same roadmap. However, we can easily blend the two theorems to bound the probability that the planner returns FAILURE for a single query by $(1-\beta)\gamma + \beta$ (this is obtained by bounding by one the probability that the planner returns FAILURE when the set of milestones is not adequate). Then, let $\alpha \in (0,1]$ be the probability of a FAILURE outcome which we are willing to tolerate. Neither the number $s$ of milestones, nor the number $g$ of trials at Step 2 of `Query-Processing` grow faster than $\log(\alpha^{-1})$, when $\alpha \rightarrow 0$.

We now turn to the description of `Permeation` and its analysis. Our goal is to evaluate how many times the complex planner must be invoked at Step 4 for the preprocessing algorithm.

`Permeation` must determine which milestones in $V$ are reachable from each other, i.e., it must partition $V$ into subsets such that all milestones in the same subset belong to the same component of $\mathcal{C}_{free}$ and no two milestones in two different subsets are in the same component of $\mathcal{C}_{free}$. If $p$ is the size of $V$, it can do so with $O(p^2)$ invocations of the complex planner by trying it on every pair of milestones in $V$, but we show below that far fewer invocations may suffice.

We work with the following abstract version of the permeation problem. The input is a graph $N$ with $p$ vertices, consisting of $k$ disjoint cliques. The goal is to determine this clique partition of $N$. The graph is presented as an adjacency matrix and the cost of an algorithm is measured by the number of entries it examines in the adjacency matrix of $N$. This is the *edge probe model* used in the study of evasive graph properties [33]. The vertices of $N$ correspond to the milestones in $V$, and an edge is present between two vertices if the corresponding milestones lie in the same component of $\mathcal{C}_{free}$. The milestones from any particular component of $\mathcal{C}_{free}$ will form a clique in $N$, and there are no edges between two distinct cliques. A probe into the adjacency matrix corresponds to an invocation of the complex planner.

Let $N(p,k)$ denote the non-deterministic complexity of this problem. A non-deterministic algorithm is only required to verify that some partition into $k$ cliques is the right partition. The non-deterministic complexity of the problem is clearly a lower bound on its deterministic and even randomized complexity.

**Theorem 3** *For $1 \leq k \leq p$, $N(p,k) = \Theta(p+k^2)$.*

We now characterize the worst-case deterministic complexity of this problem, denoted $T(p,k)$. Consider the following deterministic algorithm: by probing all edge slots incident on an arbitrary vertex $x$, determine the neighborhood of $x$, say $\Gamma(x)$; let $C_x = \{x\} \cup \Gamma(x)$, and output $C_x$; then, recur on the vertex-induced subgraph $N[V \setminus C_x]$. The number of levels in the recursion is $k$, since one of the $k$ cliques is removed from $N$ prior to each recursive call. The number of probes made in the process of determining each such clique is at most $p$. The total number of probes is $O(pk)$. The following permeation algorithm is an iterative version of the recursive algorithm (the nodes of $N$ are named $1, 2, \ldots, p$):

`Deterministic-Permeation`($V$):

1. Mark all vertices of $N$ as being LIVE.
2. Initialize $x \leftarrow 1$.
3. While $x \leq p$ do:
   (a) $\Gamma(x) \leftarrow \emptyset$.
   (b) For $y = x+1$ to $p$ do:
      i. If vertex $y$ is marked LIVE then probe the edge $(x,y)$ in $N$.
      ii. If edge $(x,y)$ is probed and found present then mark $y$ as DEAD and add $y$ to $\Gamma(x)$.
   (c) Output $\{x\} \cup \Gamma(x)$ as being a clique.
   (d) Mark $x$ as being DEAD.
   (e) Set $x$ to the smallest numbered LIVE vertex, or $p+1$ if there are no LIVE vertices left.

By the preceding discussion, we have:

**Theorem 4** `Deterministic-Permeation` *correctly solves the permeation problem using $O(pk)$ probes.*

The following lower bound establishes that `Deterministic-Permeation` is optimal. The proof uses a non-trivial adversary argument [23].

**Theorem 5** *For $1 \leq k \leq p$, $T(p,k) = \Omega(pk)$.*

We now give a randomized algorithm that beats the lower bound of Theorem 5 when the sizes of the $k$ cliques differ significantly, which is often the case in practice (when $k > 1$). `Randomized-Permeation` labels the vertices in a random order and then invokes `Deterministic-Permeation`.

`Randomized-Permeation`($V$):

1. Permute the vertices randomly. Rename the nodes by $1, \ldots, n$, in the order of the generated list.
2. Invoke `Deterministic-Permeation`($V$).

Let $w_1 \geq w_2 \geq \cdots \geq w_k$ be the sizes of the cliques in an instance $N$ arranged in a non-increasing order, where $p = \sum_{i=1}^{k} w_i$. Denote by $C_i$ the $i$th largest clique in $N$. Define a function $u()$ on an ordered $k$-tuple of positive integers $n_1, n_2, \ldots, n_k$ by $u(n_1, n_2, \ldots, n_k) = \sum_{i=1}^{k} i n_i$.

**Theorem 6** `Randomized-Permeation` *correctly determines the clique structure and incurs an expected cost that is at most*

$$2u(w_1, w_2, \ldots, w_k) - p - k.$$

*Furthermore, with high probability, the cost is at most*

$$O(u(w_1, w_2, \ldots, w_k) \log p).$$

Observe that the worst case is when all $w_i$ are equal to $p/k$, in which case the expected cost is $O(pk)$. On the other hand when there is one giant clique and $k-1$ cliques of size $O(1)$ the expected cost is $\Theta(p+k^2)$, which is essentially the non-deterministic lower bound.

## 4.2 Path Clearance Assumption

The visibility volume assumption does not prevent the existence of narrow passages in $\mathcal{C}_{free}$. To remove the need for the "complex planner" and get closer to the roadmap planner of Section 3.2, we now consider a seemingly stronger assumption: between the two configurations given by a query, there exists a collision-free path $\tau$ that achieves some clearance $\xi$ between the robot and the obstacles. More formally, let us parametrize $\tau$ by the arc length $\ell$ from the initial configuration and let $L$ designate the path's total length, i.e., $\tau : \ell \in [0, L] \rightarrow \tau(\ell) \in \mathcal{C}_{free}$. We define $\xi(\ell)$ to be the Euclidean distance between $\tau(\ell)$ and the free space boundary, and $\xi_{inf}$ to be $\inf_{\ell \in [0,L]} \xi(\ell)$.

We consider a variant of the roadmap planner in which the preprocessing algorithm consists of the first two steps of the algorithm given in Section 4.1. Unlike the planner of Section 3.2, it does not include the resampling step. The query-processing algorithm is also simpler than in Section 3.2, in that it only checks that the initial and goal configurations see milestones in the roadmap. If any one of these connections fails, the query-processing algorithm returns NO. Hence, this query-processing algorithm is deterministic.

Under the path clearance assumption, any NO outcome is incorrect. Let $\alpha$ be the probability that we are willing to tolerate for this event. The following two theorems relate the size of the roadmap to this probability, as well as to the two parameters of the assumption, that is, the length $L$ of the hypothesized path and its clearance. They give a performance guarantee for the whole planner.

**Theorem 7** *Let $\alpha \in (0, 1]$ be a positive real constant. Let $a$ be the constant $2^{-n}\mu(\mathcal{B}_1)/\mu(\mathcal{C}_{free})$ where $\mathcal{B}_1$ denotes the unit ball in $\mathbb{R}^n$. If $s$ is chosen such that:*

$$\frac{2L}{\xi_{inf}}(1 - a\xi_{inf}^n)^s \leq \alpha, \tag{1}$$

*then the planner outputs* YES *with probability at least $1 - \alpha$.*

Note that, for any given $L$ and $\xi_{inf}$, the quantity on the left side of the above inequality tends toward zero when $s \rightarrow \infty$. It is even more important to remark that it depends exponentially on $s$, so that the number of milestones $s$ needed grows no faster than the logarithm of $1/\alpha$.

The following theorem is similar to the previous one, but makes use of the clearance distribution $\xi(\ell)$ rather than just its infimum:

**Theorem 8** *Let $\alpha \in (0, 1]$ be a positive real constant. Let $a$ be the same constant as in Theorem 7. If $s$ is chosen such that:*

$$6 \int_0^L \frac{(1 - (a/2^n)\xi^n(\ell))^s}{\xi(\ell)} d\ell \leq \alpha, \tag{2}$$

*then the planner outputs* YES *with probability at least $1 - \alpha$.*

Using the inequality $1 - x \leq e^{-x}$, for $x \geq 0$, we get the following easier-to-use relations:

- The bound of Theorem 7 becomes:

$$\frac{2L}{\xi_{inf}} \exp(-a\xi_{inf}^n s) \leq \alpha. \tag{3}$$

- The bound of Theorem 8 becomes:

$$6 \int_0^L \frac{\exp(-a2^{-n}\xi^n(\ell)s)}{\xi(\ell)} d\ell \leq \alpha. \tag{4}$$

Relation 3 implies that the number of milestones that the planner must generate to output YES with probability at least $1 - \alpha$ is polynomial in $1/\xi_{inf}$ and logarithmic in $L$.

Remark that $\xi(\ell) \geq (1/2\rho)$ CLEARANCE$(\tau(\ell))$, where $\rho$ is the constant introduced in Section 2. Indeed, for any $\boldsymbol{q} \in \mathcal{C}_{free}$, let $\boldsymbol{q}^c$ be the configuration in the free space boundary that achieves minimum distance with $\boldsymbol{q}$ and let $\xi(\boldsymbol{q})$ be this distance. We have: $\xi(\boldsymbol{q}) \geq \max_i |q_i - q_i^c|$. By the definition of $\rho$, all configurations $\boldsymbol{q}'$ such that $\max_i |q_i - q_i'| \leq$ CLEARANCE$(\boldsymbol{q})/2\rho$ are in the free space. Hence, CLEARANCE$(\boldsymbol{q})/2\rho < \max_i |q_i - q_i^c| \leq \xi(\boldsymbol{q})$. Therefore, the bounds given above remain valid if we choose to define $\xi(\ell)$ as $(1/2\rho)$ CLEARANCE$(\tau(\ell))$, rather than as the distance between $\tau(\ell)$ and the free space boundary.

## 4.3 Example

Consider the two-dimensional problem ($n = 2$) shown in Figure 1. This problem is parametrized by $\omega$. The walls are obstacles, but have zero thickness; hence, $\mu(\mathcal{C}_{free}) = 1$. We are interested in the order of magnitude of $s$ when $\omega$ is small. Let $x \asymp y$ mean that $C^{-1}x \leq y \leq Cx$ for some constant $C > 0$. In particular, we have $L \asymp 1$ and $\xi_{inf} \asymp \omega$.

**Estimate of $s$ using Theorem 1:** The space in Figure 1 is $\epsilon$-good for $\epsilon = 2\omega^2$ (clearly, because of the "box" on the left). In order to bound the probability that the query-processing algorithm

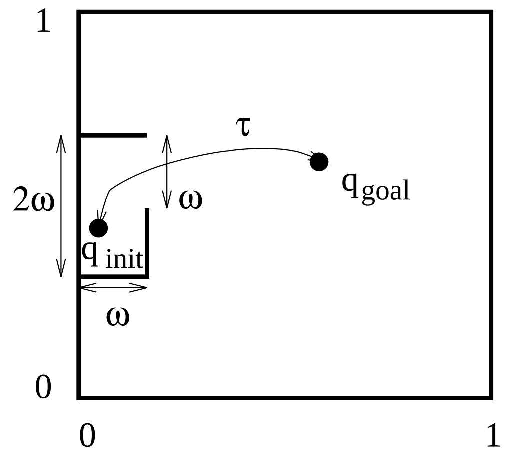


Figure 1: Illustrative example

of section 4.1 outputs FAILURE, while a collision-free path exists (Theorem 2), the roadmap must be adequate. According to Theorem 1, this is achieved with high probability if:

$$s \asymp \frac{1}{\omega^2} \log \frac{1}{\omega}. \qquad (5)$$

**Estimate of $s$ using Theorem 7:** Using (3) we can derive the following order of magnitude for $s$ that will make the planning algorithm of Section 4.2 to return YES with high probability [19]:

$$s \asymp \frac{1}{\omega^2} \log \frac{1}{\omega}. \qquad (6)$$

**Estimate of $s$ using Theorem 8:** The bound given by (4) leads to choosing (see [19]):

$$s \asymp \frac{1}{\omega^2} \log\log \frac{1}{\omega}. \qquad (7)$$

**Comparison:** Note that to answer queries with high probability it is necessary and sufficient in this example to pick a bounded number of milestones in the box, which happens with probability $\asymp \omega^2$. Hence, a tight estimate of $s$ is

$$s \asymp \frac{1}{\omega^2}.$$

The estimates (5), (6), and (7) can be seen as the unavoidable quantity $1/\omega^2$ times some factor. By exploiting the fact that $\xi(\ell)$ is small only briefly, we get a smaller factor in (7) than in (6).

# 5 Discussion

Between complete planners, which take prohibitive time to run in large dimensional configuration spaces, and ad hoc planners, which are too unreliable, there exist planners that are reasonably efficient while achieving some form of completeness. In this paper we have discussed a class of such planners. These are based on a general computational scheme presented in Section 2, which consists of randomly sampling the free space and checking pairs of sample configurations for adjacency. By choosing an appropriate sampling strategy, this scheme can be turned into a concrete planner that achieves probabilistic completeness. In Section 3 we have presented two examples of probabilistically complete planners: the potential-field and the roadmap planners. In Section 4 we have analyzed in more detail the probabilistic completeness of variants of the roadmap planner, under two different assumptions: the visibility volume assumption and the path clearance assumption. In each case, we have established a relation between the probability that the planner finds a path, when one exists, and its running time (measured in that case by the number of milestones in the roadmap). Both analyses show that the number of milestones needed grows only as the absolute value of the logarithm of the probability of an incorrect answer that we are willing tolerate. Moreover, this number is polynomial in the inverse of the parameter used to measure the goodness of free space ($\epsilon$ in the first analysis, $\xi_{inf}$ in the second).

How realistic and useful are our assumptions? Clearly, in most cases, there is no pratical way to verify that they are satisfied. Furthermore, it is easy to create spaces that are $\epsilon$-good for very small values of $\epsilon$, as well as spaces that are not $\epsilon$-good for any positive value of $\epsilon$. For instance, this is the case of the space comprised between two circles $C_1$ and $C_2$ tangent at some point $P$, with $C_2$ contained in $C_1$. On the other hand, in many applications, we are willing to discard collision-free paths if their clearance is too small, due to uncertainties in robot control and sensing. In those cases, a planner that would return a path with high probability, whenever there exists one that has enough clearance, would be quite satisfactory.[10] Remark also that, if there exists a collision-free path with a clearance greater than some given $\xi_{inf}$, this path must lie in a subset of $\mathcal{C}_{free}$ that is at least $\epsilon$-good for an $\epsilon$ that can be derived[11] from $\xi_{inf}$. Hence, the assumption of $\epsilon$-

[10] Note that the roadmap planner may nevertheless return a path with a smaller clearance.

[11] However, the bound derived from $\xi_{inf}$ may be much smaller than the actual $\epsilon$. E.g., if the free space is a rectangle having very small width $\xi$, then $\xi_{inf} \leq \xi$, but $\epsilon = 1$.

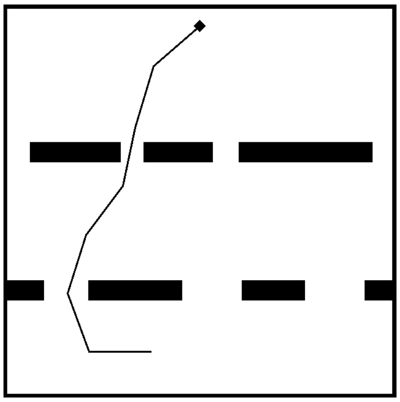

Figure 2: Robot arm example

goodness is realistic as well: for any given value of $\epsilon$, our analysis characterizes the probability that the planner finds a path in the subset of $\mathcal{C}_{free}$ that is $\epsilon$-good.

But we think that much more could be done. In particular, none of the two analyses we have given considers the resampling step (Step 4) in the preprocessing algorithm of Section 3.2. On the other hand, our experiments have shown that adding this step yields a much better roadmap planner. In relation to this observation, we have empirically tested the free space corresponding to the setting of Figure 2 (a seven-revolute-dof planar arm among several barriers forming multiple gates) for $\epsilon$-goodness. To do so we have picked 9,000 milestones at random and we have computed how many other milestones each milestone can see. The milestones with the "most" visibility could only see about 0.06 (i.e., 6%) of the remaining milestones, suggesting that they are 0.06-good. As many as 3.3% of the milestones could see no other milestones, and fully 22% could see 0.001 (i.e., 0.1%) or less; in other words, only about 78% of the configuration space is 0.001-good or better. On the other hand, the implemented planner, which includes the resampling step, handles queries in this setting with quasi-perfect reliability, after having constructed a roadmap containing 5,000 to 10,000 milestones. Other experiments confirm that the use of `Resampling` improves the coverage of the free space by the milestones.

To explain the role of `Resampling` we introduce a generalization of the notion of $\epsilon$-goodness. We say that a free configuration $\boldsymbol{q}$ is $(\epsilon, 1)$-good if $\mu(\mathcal{S}(\boldsymbol{q})) \geq \epsilon\mu(\mathcal{C}_{free})$, corresponding to our original definition of $\epsilon$-goodness for a configuration. Next, we say a free configuration $\boldsymbol{q}$ is $(\epsilon, t)$-good if $\mu(\{\boldsymbol{q}' \in \mathcal{S}(\boldsymbol{q}) \mid \boldsymbol{q}' \text{ is } (\epsilon, t-1)\text{-good}\}) \geq \mu(\mathcal{S}(\boldsymbol{q}))/2$. For $t > 1$, we say that $\mathcal{C}_{free}$ is $(\epsilon, t)$-good if $\mu(\{\boldsymbol{q} \in \mathcal{C}_{free} | \boldsymbol{q} \text{ is } (\epsilon, 1)\text{-good}\}) \geq \mu(\mathcal{C}_{free})/2$ and every free configuration is $(\epsilon, i)$-good for $i \leq t$. If $\mathcal{C}_{free}$ is $(\epsilon, t)$-good for a small value of $t$, we can give a theoretical basis for the resampling strategy. The main idea is that single links discovered by the connector in the algorithms are now simulated using $t$-link paths found by resampling and connecting using the connector. This leads to a generalized definition of an adequate set of milestones, and eventually to a version of Theorem 2 in which the number of invocations of the connector is larger by a factor of $2^t$. See [23] for more detail.

We believe that future research should develop and analyze new sampling strategies, as well as combinations of strategies, under various assumptions. In particular, it would be of particular interest to investigate strategies suitable for single-query planning problems. In this case, instead of spreading milestones all over the free space, one could expand a "cloud" of sample configurations from the initial configuration toward the goal, or in both directions concurrently. This could be achieved by iteratively generating configurations in small neighborhoods of previously generated configurations, while keeping only those configurations which are in the same free space component as the initial configuration. This strategy could use a function similar to a potential field as a probability distribution to randomly choose which configurations to expand from at every iteration. In the multi-query case, it would also be useful to consider the case where all queries are made in the same component of $\mathcal{C}_{free}$ and the preprocessing is given one configuration in that component. The case where the volume of the free space is very small relative to the total volume of the configuration space should also be investigated. We envision that ultimately randomized planners will automatically select and combine sampling strategies from a library of basic strategies to closely match the characteristics of the input problems. This sort of combination has already given good results in another area, graphic rendering [46].

We did not discuss distance calculation in this paper, but it is a key computation in the context of our sampling scheme. Indeed, our planners spend most of their time evaluating CLEARANCE. So, any improvement of the time efficiency of the algorithm implementing CLEARANCE would translate into a planning improvement of the same order of magnitude. One could also try to match the distance calculation algorithm and the sampling strategy, to get the best global result. For example, a sampling strategy generating configurations in small neighborhoods of previously generated configurations could be combined with an incremental distance calculation algorithm such as the one proposed in [32]. Such an algorithm uses the nearest points computed at the previous configuration to quickly find the nearest points at the new configuration. As another example, the distance calculation algorithm proposed in [39] (and used in our planners) precomputes data structures to later accelerate the processing of distance calculation queries. Each such data structure is a binary tree of spheres approximating a single rigid body. Given two bodies, a simple traversal of their respective trees allows us to quickly detect collision or identify the nearest points between the two bodies. A perhaps more efficient solution would be to create a single data structure for all the rigid bodies forming the robot and to efficiently update this data structure when configuration parameters vary (as proposed in [12]). Then the sampling strategy should try to generate configurations in a sequence that minimizes update costs.

The random sampling scheme for path planning proposed in this paper offers many opportunities for further research. We are confident that it can ultimately produce very fast planners able to correctly handle a wide range of problems with high probability.

**Acknowledgments:** The development of the potential-field and roadmap planners has been supported by ARPA contracts DAAA21-89-C0002, N00014-92-J-1809, and N00014-94-1-0721. The formal analysis of these planners and the design of the general random sampling scheme was supported by MURI grant DAAH04-96-1-0007 (Army). The experimental part of this research has also benefited from gifts by General Electric, General Motors, Renault, and Rockwell. Lydia Kavraki and Jean-Claude Latombe are partially supported by ARPA contract N00014-94-1-0721 and a MURI grant of the Army. Rajeev Motwani is supported by an Alfred P. Sloan Research Fellowship, an IBM Faculty Partnership Award, and NSF Young Investigator Award CCR-9357849, with matching funds from IBM, Schlumberger Foundation, Shell Foundation, and Xerox Corporation. This paper has benefited from useful comments by Leo Guibas, Dan Halperin, David Hsu, and Jean-Paul Laumond.

# References

[1] J.M. Ahuactzin Larios, *Le Fil d'Ariane: Une Méthode de Planification Générale. Application à la Planification Automatique de Trajectoires*, Thèse de l'Institut National Polytechnique de Grenoble, Septembre 1994.

[2] S. Arya and D.M. Mount, Approximate Range Searching, *Proc. of ACM Symp. on Computational Geometry*, 172-181, 1995.

[3] J. Barraquand and J.C. Latombe, Robot Motion Planning: A Distributed Representation Approach, *The Int. J. of Robotics Research*, 10(6):628-649, 1991.

[4] J. Barraquand, B. Langlois, and J.C. Latombe, Numerical potential Field Techniques for Robot Path Planning, *IEEE Tr. on Syst., Man, and Cyb.*, 22(2):224-241, 1992.

[5] P. Bessière, E. Mazer, J.M. Ahuactzin, Planning in a Continuous Space with Forbidden Regions: The Ariadne's Clew Algorithm, *Algorithmic Foundations of Robotics*, K. Goldberg et al. (eds.), AKPeters, Wellesley, MA, 39-47, 1995.

[6] J.F. Canny, *The Complexity of Robot Motion Planning*, MIT Press, Cambridge, MA, 1988.

[7] D. Challou and M. Gini, Parallel Formulation of Informed Randomized Search for Robot Motion Planning Problems, *Proc. of IEEE Int. Conf. on Robotics and Automation*, 709-714, Nayoga, Japan, 1995.

[8] H. Chang and T.Y. Li, Assembly Maintainability Study with Motion Planning, *Proc. of IEEE Int. Conf. on Robotics and Automation*, 1012-1019, Nagoya, 1995.

[9] P.C. Chen, Adaptive Path Planning: Algorithm and Analysis, *Proc. IEEE Int. Conf. on Robotics and Automation*, 721-728, Nagoya, 1995.

[10] E.G. Gilbert, D.W. Johnson, and S.S. Keerthi, A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space, *IEEE J. of Robotics and Automation*, 4(2):193-203, 1988.

[11] L. Graux, P. Millies, P.L. Kociemba, and B. Langlois, Integration of a Path Generation Algorithm into Off-Line Programming of AIRBUS Panels. *Aerospace Automated Fastening Conf. and Exp.*, SAE Tech. Paper 922404, 1992.

[12] D. Halperin, J.C. Latombe, and R. Motwani, *Dynamic Maintenance of Kinematic Structures*, Tech. Rep. , Dept. of Computer Science, Stanford Univ., 1995.

[13] J.E. Hopcroft, J.T. Schwartz, and M. Sharir, On the Complexity of Motion Planning for Multiple Independent Objects: PSPACE-Hardness of the 'Warehouseman's Problem', *Int. J. of Robotics Research*, 3(4):76-88, 1984.

[14] J.E. Hopcroft and G.T. Wilfong, Reducing Multiple Object Motion Planning to Graph Searching. *SIAM J. on Computing*, 15(3):768-785, 1986.

[15] T. Horsch, F. Schwarz, and H. Tolle, Motion Planning for Many Degrees of Freedom - Random Reflections at C-Space Obstacles, *Proc. of IEEE Int. Conf. on Robotics and Automation*, 3318-3323, San Diego, CA, 1994.

[16] Y.K. Hwang and P.C. Chen, A Heuristic and Complete Planner for the Classical Mover's Problem, *Proc. IEEE Int. Conf. on Robotics and Automation*, 729-736, Nagoya, 1995.

[17] D.A. Joseph and W.H. Plantiga, On the Complexity of Reachability and Motion Planning Questions. *Proc. of the First ACM Symp. on Computational Geometry*, 62-66, 1985.

[18] L. Kavraki, *Random Networks in Configuration Space for Fast Path Planning*, Ph.D. Thesis, Rep. STAN-CS-TR-95-1535, Comp. Sci. Dept., Stanford Univ., January 1995.

[19] L. Kavraki, M. Kolountzakis, and J.C. Latombe, Analysis of Probabilistic Roadmaps for Path Planning, manuscript submitted to *IEEE Int. Conf. on Robotics and Automation*, 1996.

[20] L. Kavraki and J.C. Latombe, Randomized Preprocessing of Configuration Space for Fast Path Planning, *Proc. IEEE Int. Conf. on Robotics and Automation*, 2138-2145, San Diego, CA, 1994.

[21] L. Kavraki and J.C. Latombe, Randomized Preprocessing of Configuration Space for Path Planning: Articulated Robots, *Proc. IROS*, 1764-1771, München, Germany, 1994.

[22] L. Kavraki, J.C. Latombe, R. Motwani, and P. Raghavan, *Randomized Query Processing in Robot Motion Planning*, Rep. STAN-CS-TR-94-1533, Dept. of Computer Science, Stanford University, 1994.

[23] L. Kavraki, J.C. Latombe, R. Motwani, and P. Raghavan, Randomized Query Processing in Robot Path Planning, *27th Annual ACM Symp. on Theory of Computing (STOC)*, 353-362, Las Vegas, NV, 1995.

[24] L. Kavraki, P. Švestka, J.C. Latombe, and M. Overmars, Probabilistic Roadmaps for Fast Path Planning in High Dimensional Configuration Spaces, to appear in *IEEE Tr. on Robotics and Automation*.

[25] D.E. Koditschek, Exact Robot Navigation by Means of Potential Functions: Some Topological Considerations, *Proc. IEEE Int. Conf. on Robotics and Automation*, Raleigh, NC, 1-6, 1987.

[26] Y. Koga, K. Kondo, J. Kuffner, and J.C. Latombe, Planning Motions with Intentions. *Proc. of SIGGRAPH'94*, 395-408, 1994.

[27] Y. Koga and J.C. Latombe, On Multi-Arm Manipulation Planning, *Proc. IEEE Int. Conf. on Robotics and Automation*, 945-952, San Diego, CA, 1994.

[28] F. Lamiraux and J.P. Laumond, *On the Expected Complexity of Random Path Planning*, Rep. No. 95087, LAAS/CNRS, Toulouse, March 1995.

[29] J.C. Latombe, *Robot Motion Planning*, Kluwer Academic Publ., Boston, MA,1991.

[30] J.C. Latombe, Controllability, Recognizability, and Complexity Issues in Robot Motion Planning, *Proc. 36th Annual Symp. on Foundations of Computer Science (FOCS)*, 484-500, Milwaukee, Wisconsin, 1995.

[31] T.Y. Li and J.C. Latombe, On-Line Motion Planning for Two Robot Arms in a Dynamic Environment, *Proc. IEEE Int. Conf. on Robotics and Automation*, 1048-1055, Nagoya, 1995.

[32] M.C. Lin and J.F. Canny, A Fast Algorithm fir Incremental Distance Calculation, *Proc. IEEE Int. Conf. on Robotics and Automation*, 1008-1014, Sacramento, CA, 1991.

[33] L. Lovász and N. Young, *Lecture Notes on Evasiveness of Graph Properties*, Tech. Rep. CS-TR-317-91, Comp. Sci. Dept., Princeton Univ., 1991.

[34] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.

[35] M. Overmars, *A Random Approach to Motion Planning*, Tech. Rep. RUU-CS-92-32, Dept. of Comp. Sci., Utrecht Univ., 1992.

[36] M. Overmars and P. Švestka, A Probabilistic Learning Approach to Motion Planning, *Algorithmic Foundations of Robotics*, K. Goldberg et al. (eds.), A.K.Peters, Wellesley, MA, 19-37, 1995.

[37] M. Ponamgi, D. Manocha, and M.C. Lin, Incremental Algorithms for Collision Detection Between Solid Models, *Proc. 3rd ACM Symp. on Solid Modeling and Applications*, 293-304, Salt Lake City, UT, 1995.

[38] F.P. Preparata and M.I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, NY, 1985.

[39] S. Quinlan, Efficient Distance Computation Between Non-Convex Objects. *Proc. IEEE Int. Conf. on Robotics and Automation*, 3324-3330, San Diego, CA, 1994.

[40] J. Reif, Complexity of the Mover's Problem and Generalizations. *FOCS*, 421-4127, 1979.

[41] J.H. Reif and M. Sharir, Motion Planning in the Presence of Moving Obstacles, *proc. 25th IEEE Symp. on Foundations of Computer Science*, 144-154, 1985.

[42] J.T. Schwartz and M. Sharir, On the 'Piano Movers' Problem: II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds. *Advances in Applied Mathematics*, 4:298-351, 1983.

[43] S. Sekhavat, P. Švestka, J.P. Laumond, and M. Overmars, *Probabilistic Path Planning for Tractor-Trailer Robots*, Tech. Rep., LAAS/CNRS, Toulouse, September 1995.

[44] P. Švestka, *A Probabilistic Approach to Motion Planning for Car-Like Robots*, RUU-CS-93-18, Comp. Sci. Dept., Utrecht Univ., The Netherlands, April 1993.

[45] F. Thomas and C. Torras, Interference Detection Between Non-Convex Polyhedra Revisited with a Practical Aim, *Proc. IEEE Int. Conf. on Robotics and Automation*, 587-590, San Diego, CA, 1994.

[46] E. Veach and L.J. Guibas, Optimally Combining Sampling Techniques for Monte Carlo Rendering, *Proc. of SIGGRAPH'95*, 419-428, 1995.

[47] X. Zhu and K. Gupta, *On Local Minima and Random Search in Robot Motion Planning*, Technical Report, Simon Fraser Univ., BC, Canada, 1993.

# Appendix: Selected Proofs

## A. Proof of Theorem 1

Let $M$ denote the set of the $s$ randomly chosen milestones. The volume $H$ of configurations not visible from any of these milestones is:

$$\mu(\{\boldsymbol{q} \in \mathcal{C}_{free} \mid \boldsymbol{q} \notin \cup_{\boldsymbol{m} \in M} \mathcal{S}(\boldsymbol{m})\}).$$

Its expected value is:

$$E[H] = \int_{\boldsymbol{q} \in \mathcal{C}_{free}} \Pr[\boldsymbol{q} \notin \cup_{\boldsymbol{m} \in M} \mathcal{S}(\boldsymbol{m})] d\boldsymbol{q}.$$

The $\epsilon$-goodness of the free space entails that the probability that any given configuration is not visible from any of the $s$ milestones is at most $(1-\epsilon)^s$. Thus:

$$E[H] \leq \mu(\mathcal{C}_{free})(1-\epsilon)^s \leq \mu(\mathcal{C}_{free})\epsilon\beta/4. \quad (8)$$

Given a random variable $X$ assuming only non-negative values, the Markov inequality [34]:

$$\Pr[X \geq x] \leq E[X]/x$$

holds for all $x \in \mathbb{R}^+$. Using this inequality and the relation (8), we get:

$$\Pr[H > (\epsilon/2)\mu(\mathcal{C}_{free})] \leq \beta/2.$$

Hence, with probability $1-\beta/2$, $H$ is at most $(\epsilon/2)\mu(\mathcal{C}_{free})$, in which case $M$ is adequate.

## B. Proof of Theorem 2

Assume an adequate set of milestones. For any $\boldsymbol{q} \in \mathcal{C}_{free}$, the volume of the subset of $\mathcal{S}(\boldsymbol{q})$ visible from some milestone is at least:

$$\mu(\mathcal{S}(\boldsymbol{q})) - (\epsilon/2)\mu(\mathcal{C}_{free}) \geq (\epsilon/2)\mu(\mathcal{C}_{free}).$$

Therefore, for either query configuration $\boldsymbol{q}_i$ ($i \in \{\text{init}, \text{goal}\}$), the probability that a random configuration chosen from $\mathcal{S}(\boldsymbol{q}_i)$ is not visible from any milestone is at most $1/2$. The probability that `query-processing` fails to connect $\boldsymbol{q}_i$ to a milestone on $\log(2/\gamma)$ trials at Step 1(b)i is thus less than $\gamma/2$. Since Step1(b)i is performed for both query configurations, the overall failure probability is at most $\gamma$.

## C. Proof of Theorem 7

We assume the existence of a path $\tau : \ell \in [0, L] \mapsto \tau(\ell) \in \mathcal{C}_{free}$ connecting $\boldsymbol{q}_{init}$ to $\boldsymbol{q}_{goal}$, where $\ell$ stands for the arc length from $\boldsymbol{q}_{init}$. For simplicity, let $\xi$ (instead of $\xi_{inf}$) designate the infimum of the Euclidean distance between $\tau(\ell)$ and the free space boundary, when $\ell$ spans the interval $[0, 1]$. Given any two configurations $\boldsymbol{q} = \tau(\ell)$ and $\boldsymbol{q}' = \tau(\ell')$ on $\tau$, let $l(\boldsymbol{q}, \boldsymbol{q}')$ denote the path length $|\ell - \ell'|$.

Let $\mathcal{B}_r(x)$ designate the ball of radius $r$ centered at $x \in \mathbb{R}^n$. We pick $k = \lceil 2L/\xi \rceil$ configurations on $\tau$, denoted by $\boldsymbol{q}_0 = \boldsymbol{q}_{init}, \boldsymbol{q}_1, \ldots, \boldsymbol{q}_k = \boldsymbol{q}_{goal}$, such that $l(\boldsymbol{q}_i, \boldsymbol{q}_{i+1}) \leq \xi/2$, for all $i \in [0, k-1]$. We have that:

$$\mathcal{B}_{\xi/2}(\boldsymbol{q}_{i+1}) \subset \mathcal{B}_{\xi}(\boldsymbol{q}_i).$$

For any two points $p_i \in \mathcal{B}_{\xi/2}(\boldsymbol{q}_i)$ and $p_{i+1} \in \mathcal{B}_{\xi/2}(\boldsymbol{q}_{i+1})$, the straight-line segment connecting $p_i$ and $p_{i+1}$ lies entirely in $\mathcal{C}_{free}$; indeed, the above relation implies that $p_{i+1}$ also lies in $\mathcal{B}_{\xi}(\boldsymbol{q}_i)$. So, a sufficient condition for the query-processing algorithm to find a path is that each ball $\mathcal{B}_{\xi/2}(\boldsymbol{q}_i)$, $i = 1, \ldots, k-1$ contains at least one milestone. The probability that a ball of radius $r$ lying entirely in the free space contains none of the $s$ milestones is $(1 - \mu(\mathcal{B}_r)/\mu(\mathcal{C}_{free}))^s$. In $\mathbb{R}^n$ we have $\mu(\mathcal{B}_r) = r^n \mu(\mathcal{B}_1)$. Therefore, the probability that the planner does not find a path is at most:

$$\left(\lceil \frac{2L}{\xi} \rceil - 1\right) \left(1 - \frac{\mu(\mathcal{B}_{\xi/2})}{\mu(\mathcal{C}_{free})}\right)^s,$$

which is itself no greater than:

$$\frac{2L}{\xi} \left(1 - 2^{-n} \frac{\mu(\mathcal{B}_1)}{\mu(\mathcal{C}_{free})} \xi^n\right)^s.$$

Hence, choosing $s$ such that the above quantity is at most $\alpha \in (0, 1]$ guarantees that the planner will find a path with probability at least $1 - \alpha$.