LOCAL RANDOMIZATION IN NEIGHBOR SELECTION IMPROVES PRM ROADMAP QUALITY

A Thesis

by

BRYAN CLAY BOYD

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2012

Major Subject: Computer Science

ABSTRACT

Probabilistic Roadmap Methods (PRMs) are one of the most used classes of motion planning methods. These sampling-based methods generate robot configurations (nodes) and then connect them to form a graph (roadmap) containing representative feasible pathways. A key step in PRM roadmap construction involves identifying a set of candidate neighbors for each node. Traditionally, these candidates are chosen to be the k-closest nodes based on a given distance metric. This work proposes a new neighbor selection policy called $LocalRand(k, k')$, that first computes the k' closest nodes to a specified node and then selects k of those nodes at random. Intuitively, $LocalRand$ attempts to benefit from random sampling while maintaining the higher levels of local planner success inherent to selecting more local neighbors. A methodology for selecting the parameters k and k' is provided, and an experimental comparison for both rigid and articulated robots show that $LocalRand$ results in roadmaps that are better connected than the traditional k-closest or a purely random neighbor selection policy. The cost required to achieve these results is shown to be comparable to the cost of k-closest.

To my wife, Michelle

ACKNOWLEDGMENTS

This work represents a great deal of support and help from many people:

First, I must thank my advisor, Prof. Nancy Amato. Graduate research was not even a consideration until I was offered the chance to participate in undergraduate research under her guidance and mentorship. She has helped me grow tremendously throughout my time in the Parasol Lab as a researcher, team member, and leader; without her influence and guidance, this work would not be possible. I will always be grateful for her encouragement, patience, and constant willingness to help.

I would like to thank the members of my committee: Dr. Takis Zourntos and Dr. Jennifer Welch. Thank you for your feedback, cooperation and help.

I am deeply grateful for the help of the former and current members of the Parasol Lab. Roger Pearce has been a terrific mentor over the course of my studies: in guiding my undergraduate research, providing collaboration on papers, and being an invaluable coding resource. Thanks for taking time out to help with my many, many questions. Dr. Shawna Thomas and Dr. Lydia Tapia have provided valuable help through the years with research opportunities early on and guidance in my own projects. I also must thank Troy McMahon and Sam Jacobs for their ideas, hard work, and friendship. I have truly enjoyed working with you both on our many collaborations.

Thank you to my family and friends for your support and encouragement throughout my studies. I especially thank my father, who instilled in me a love of computer science at an early age and helped me discover that work can still be play.

Finally, I must thank my lovely wife Michelle for her unwavering love, support and understanding during my studies. Her french-press coffee, baked goods, and presence helped make the long hours of research enjoyable. I am truly blessed to have her in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

The general *motion planning* problem involves finding a valid path for an object (e.g. robot, vechicle) from some start to goal configuration in a given environment. The notion of validity depends on the problem – traditionally, it refers to a collision-free path, i.e. avoiding collisions with self and with obstacles in the environment. Motion planning is an important component of many applications, including computer-aided design [1], robotics [2], virtual reality simulations [3], and bioinformatics [4] .

The motion planning problem is regarded intractable, as the complexity of an exact algorithm grows exponentially in the complexity of the robot [5]. Research on randomized, sampling-based approaches has produced methods that can solve important motion planning problems that were once considered impractical. One widely used randomized method is the Probabilistic Roadmap Method (PRM) [6]. PRMs sample and test motions in the space consisting of robot configurations, called *configuration space* (C-space).

PRMs use a *roadmap graph* $G = (V, E)$ to approximate the environment in C-space, where vertices $v \in V$ represent configurations in C-space and edges $e \in E$ are valid paths between them. In implementations of PRMs, valid configurations are selected through a *sampling method* and connected using a *connection strategy* and *local planner*. Queries are solved by connecting the start and goal configuration to the roadmap and using a graph search (e.g. Djikstra's shortest-path algorithm) to extract a path.

One of the key steps in PRM construction is node connection. Ideally, roadmap connectivity should reflect the connectivity of the underlying C-space. From this perspective, the best strategy would be to attempt to connect all $\theta(n^2)$ pairs of nodes. However, the enormous cost of these connection attempts is not feasible for any but the simplest of problems. Hence, the selection of candidates for local transitions (neighbors) is crucial to both roadmap quality and efficiency.

The objective of a good neighbor selection strategy is to identify pairs of configuration that have a high probability of being connectible by the local planner and that are useful in terms of producing good quality roadmaps.

The most commonly used method for neighbor selection in PRMs uses nearest-neighbor search to select the k nodes that are closest to the node in question, where k is typically some relatively small, fixed constant, typically between 5 and 25 [3].

This work proposes a new neighbor selection policy called $LocalRand(k, k')$. This method first computes the k' closest nodes to a specified node and then selects k of those nodes at random; this enables us to limit the proximity of nodes from which neighbors are selected while still allowing randomness in selection.

Intuitively, the proposed $LocalRand$ method attempts to achieve the benefits associated with random sampling, while also maintaining the higher levels of local planner success associated with local connection. Additionally, basic differences between $LocalRand$ and the traditional k-closest neighbor selection policy are expected to affect roadmap structure. For example, in undirected roadmaps $LocalRand$ should reduce the number of duplicate connection attempts, resulting in roadmaps with more edges. Another difference is that $LocalRand$ will likely produce roadmaps with longer edges than k-closest. This may be beneficial as it has previously been shown that longer edges have a positive impact on roadmap quality [7].

A. Contributions

The main contributions of this work include:

- A new neighbor selection policy, $LocalRand(k, k')$, that identifies a set of k' local nodes and then selects a random subset of k of these nodes, and a methodology for selecting the parameters for $LocalRand$.

- An experimental evaluation that shows $LocalRand$ is capable of producing roadmaps with higher connectivity than k-closest at a comparable cost.

The performance of $LocalRand(k, k')$ is closely tied to the selection of parameters k and k'. For a given k and n (total number of configurations) we have $k \leq k' \leq n - 1$. Note that for the extreme values of $k' = k$ and $k' = n - 1$, the method becomes equivalent to k-closest and purely random (k-random) neighbor selection, respectively. To understand how to select k' for a given k and n, an extensive performance study was undertaken over the range of possible k and k' values in some basic environments. From this study, a set of optimal k' values was identified for which the $LocalRand$ method is capable of outperforming both the k-closest and k-random methods; these optimal k' varied across different environment classifications. Next, a set of more complex environments is considered, with the (k, k') values determined from the closest-matching basic environment. The results show that the $LocalRand$ method with these k' values is capable of outperforming both the k-closest and k-random methods in more complex environments including cluttered and narrow passage environments with both rigid body robots and fixed/free base articulated linkage robots of varying size and dimensionality.

B. Organization of Thesis

The thesis is organized as follows. In chapter II, we provide an introduction to the Probabilistic Roadmap Method (PRM) and discuss related literature, especially around the area of candidate neighbor selection for PRM connection. Next (chapter III), we compare several neighbor selection strategies for the connection phase of PRM construction, and introduce the $LocalRand$ method. Chapter IV provides a description of the environments and other PRM parameters (e.g. distance metric) that will be studied during our evaluation of neighbor selection strategies. The final chapters (V, VI) detail the results of this study and conclusions drawn from the observed data.

CHAPTER II

PRELIMINARIES AND RELATED WORK

Here, we present an overview of the Probabilistic Roadmap Method and its use for solving complex motion planning problems in high-DOF configuration space. Following this, we describe variations of PRM node generation, then discuss relevant related work in PRM, especially around candidate neighbor selection.

A. Configuration Space

In the context of a motion planning problem, a robot is an object with position and orientation described by a set of n degrees of freedom (DOFs), with each corresponding to a component of the object (e.g. physical positions in (x, y, z), orientations, joint angles of a linkage). The robot's configuration can be uniquely described by a point (x_1, x_2, $\cdots$, x_n) in an n-dimensional space called *configuration space* (C-space) [8]. The subset of feasible (not in collision) configurations is called *free C-space* (C-free), and the set of unfeasible configurations is the *blocked C-space* (C-obstacle). With these definitions, the motion planning problem becomes that of finding a continous path through C-free that connects some start and goal configuration. Rather than explicitly computing the boundaries of C-obstacle, we determine the feasibility of each configuration by performing a collision detection (CD) test against obstacles in the robot's natural workspace.

B. PRM Overview

The methods proposed in this work will be studied on Probabilistic Roadmap Methods [6], a class of randomized motion planners that use a roadmap graph of free robot configurations (not in collision with obstacles) to find paths between some start and goal configuration. The use of PRMs to solve a general motion planning problem is described in Alg. 1.

Roadmap creation in PRMs is typically performed in two phases. In the **node generation** phase (step 1 in Alg. 1), a set of n robot configurations in C-space is generated using a specific sampling strategy (or set of sampling strategies), then added to the roadmap graph as vertices. The **Node Connection** phase (step 2 in Alg. 1) attempts to insert edges in this graph between vertices that represent a valid motion between two configurations.

The node connection phase itself contains two separate operations. First, pairs of candidates for connection are chosen by iterating through all roadmap vertices and selecting a set of configurations to connect to, using some criteria. In most applications of PRMs, the selection criteria is simply the k configurations that are closest, in distance, to the source configuration. After these candidates are found a connection is attempted using a *local planner*, which is a computationally-expensive deterministic planner that checks for a valid path between the two configurations (e.g., a straight-line collision detection check in C-space).

The cost of these steps is listed in Alg. 1 in terms of the primitive operations:

- c_s = cost to generate a sample
- c_v = cost to validate the feasibility of sample
- c_n = cost to find a neighbor
- c_c = cost to validate a connection between two nodes (local planner call)

Several PRM variants have been proposed in literature. The following use different methods of node generation (step 1 in Alg. 1) to handle different classifications of problems.

Algorithm 1 PRMs: Probabilistic Roadmap Methods

```
1) Generate Nodes {find collision-free configurations - $O(n(c_s + c_v))$}
2) Connect Nodes {connect nodes to form roadmap - $O(kn(c_n + c_c))$}
   for each configuration:
      a) Select neighbors - $O(k * c_n)$
      b) Attempt connection between neighbor pairs - $O(k * c_c)$
3) Process Query
   a) Connect start/goal to roadmap
   b) Find path in roadmap between connection nodes
```

- **Basic PRM (PRM):** The original PRM [6] uses uniform random sampling in order to produce a distribution that is symmetry invariant. This method performs well when the environment is uniform (e.g., completely free or a uniform distribution of obstacles).

- **Obstacle-Based PRM (OBPRM):** OBPRM [9] is a PRM that samples near the boundary of C-space obstacles. The method first generates configurations that are in collision with obstacles in C-space, then computes valid configurations in random directions away from the obstacle. Of these, the valid configuration closest to the surface is kept.

- **Gaussian-Sampling PRM (GaussPRM):** GaussPRM [10] samples configurations in difficult regions by using techniques associated with blurring in image processing.

- **Medial-Axis PRM (MAPRM):** MAPRM [11] biases sampling along the medial axis (generalized Voronoi diagram) of the valid C-space. Resulting paths generally have a wide clearance from obstacles.

In this work, we primarily use PRM and OBPRM for sampling.

C. Related Work

1. PRM

PRMs have been applied to a wide range of problems [12]. In robotics, PRM has been used for path planning with mobile robots [13] [14] [15], humanoid robots [16] and reconfigurable robots [17]. They have been applied to biological problems including analysis of biological structures [18] and protein folding [4]. They have also been used in industrial automation and path planning with robotic manipulators [19] [20] [21] [22].

In [7], Geraerts and Overmars perform a reachability-based analysis of the PRM method. In this study they evaluate existing node generation and connection methods based on how well they cover environments and how connected the roadmaps they produce are. This study shows that existing methods are capable of generating node sets that cover the environment well and that the major difficulty in roadmap construction is connecting these nodes, especially in difficult narrow passage problems. The study concludes that the major hurdle in roadmap construction is not covering the environment but generating a connected roadmap. In [23], Geraerts and Overmars show experimentally that the main difficulty in PRM construction is constructing a roadmap whose connectivity represents the connectivity of C-space.

2. Candidate Selection Approaches

There have been a variety of proposed methods for identifying candidates for local transitions during the connection phase of PRM construction [24]. The most common strategy is the so-called k-closest, which selects the set of k nodes that are nearest the query sample, i.e., its k-nearest neighbors, where k is typically some small constant. This simple strategy was used in many PRMs, including the original PRM [6], OBPRM [9], and GaussPRM [10]. The intuition behind the use of the set of closest nodes is that the costs for verifying the validity of the connection are reduced and, depending on the problem, shorter connections are more likely to be collision-free [24].

Another simple method is a *Distance* method that identifies neighbors within some fixed distance of the query node. A drawback of this method is that some knowledge of the problem is required in order to determine an appropriate distance. If the distance is too big, too many connections will be attempted, resulting in a very expensive connection phase. If it is too small, then many connections will be excluded and the roadmap will be poorly connected. The original PRM implementation included a variation of the distance method with an upper bound on the number of neighbors [6]. The distance method is commonly used for grid-based PRM methods [25]. It is also commonly used in biophysical simulations where certain cutoffs are standard, e.g., a root-mean-square deviation (RMSD) of atom positions between molecular conformations [26]. Geraerts and Overmars [27, 28] include a distance method in a study of the impact of sampling, node selection/adding strategy and local planning on coverage and connectivity of PRMs. Their study shows that the k-closest method is well-suited for generating roadmaps with high connectivity.

In [23], Geraerts and Overmars explore a *Visibility*-based connection strategy. Visibility neighbors are those that are visible (connectible with a straight-line) from the node. However, this often requires special placement of the nodes as in the variant, Visibility Roadmaps [29].

Geraerts and Overmars[23] also explore a *Component*-based selection strategy. This strategy attempts k connections to each connected component [24]. Depending on the number of components and the value of k, there could be a large number of attempts.

As part of his work in [30], Boden presents a grid based neighbor selection policy. The nodes in this method are arranged in a grid in C-space and connections are attempted between adjacent nodes on the grid. If a path is not found, then additional grid nodes are introduced at a higher resolution in areas that could not be connected, and connections are attempted between adjacent nodes in the higher resolution grid.

In [31], Karaman and Frazzoli study the asymptotic behavior of solutions returned by sampling-based motion planning methods. As part of this work they present the methods PRM* and RRT*, which are shown theoretically to converge to optimal solutions as the number of nodes grow to densely cover the configuration space.

Another direction in neighbor selection that has been explored is the examination of the impact of using approximate and more efficient strategies for computing k-closest using data structures (e.g. KD-trees [32] [33], Metric/Spill Trees [34] [35], dimensionality reduction [36]) to efficiently provide solutions to the neighbor selection problem. These methods use an approximation parameter that tolerates a certain amount of error in neighbor selection.

KD-trees are a D-dimensional extension of a binary tree which provides a good model for motion planning environments with high dimensionality. The CGAL [32] and ANN [37] libraries provide a KD-tree structure with an approximate query algorithm. The parameter ϵ can be used to allow approximation, where a k-nearest-neighbor search returns k neighbors that are guaranteed to be no more than $(1 + \epsilon)$ times farther away than the exact k-th neighbor.

The *Metric-tree* data structure organizes a set of nodes in a spatial hierarchical manner [34]. The metric tree is constructed by iteratively dividing a group of nodes into two subgroups, based on the most distant nodes they are closest to. Metric trees are queried by locating the leaf of the tree containing the query nodes, then performing a backtracking step to ensure that every leaf potentially containing a nearest-neighbor is searched. *Spill-Trees* are an approximate variant of metric trees [35] that are queried using a defeatist search, which only checks the leaf containing the query nodes.

Distance-based Projection onto Euclidean Space (DPES) [36] computes the nearest-neighbors to a given nodes by projecting the metric space down to R^d space. Although the accuracy is reduced, the complexity of the computation of nearest-neighbors is also reduced, as fewer distance evaluations are required. The amount of approximation can be controlled by increasing/decreasing the dimensionality of the projection.

CHAPTER III

CANDIDATE SELECTION POLICIES

In this chapter, we compare several **selection policies** for the PRM connection phase. In addition to the traditionally used k-closest policy, we consider an $AllPairs$ policy (used as a baseline comparison for each method), a random connection policy, and our proposed novel method, $LocalRand$, which identifies a set of local nodes then selects a subset of these nodes at random.

A. Selection Policy Definitions

In our discussion, we will define selection policies that operate on a candidate set of configurations (V_c) and a source configuration (v_s). The set of all configurations in the roadmap is V, with $V_c \subseteq V$ and $v_s \in V$. Most selection policies choose a maximum of k configurations, where k is a user provided constant.

Listed below are a set of basic strategies for selecting candidates from the candidate set V_c. Note that all strategies that select a candidate set based on distance calculation (i.e., k-closest and $LocalRand$) are affected by the distance metric used.

- $AllPairs(v_s, V_c)$: Select all configurations from V_c as connection candidates for v_s.
- k-closest(v_s, V_c, k): Select the k closest configurations to v_s from V_c.
- k-random(v_s, V_c, k). Select k configurations at random from V_c.
- $LocalRand(v_s, V_c, k, k')$ Select the $k' \geq k$ closest configurations to v_s from V_c and then select k of them at random.

B. LocalRand Candidate Neighbor Selection Policy

The $LocalRand$ method first identifies the k' closest neighbors to a sample then selects k of these nodes at random. One of the key features of this method is that it allows us to introduce a controlled amount of randomness into neighbor selection. Because this method first locates the k' closest neighbors we know that the nodes selected by this method must be drawn from the k' closest nodes to a sample. By changing the value of k' we can change the range from which the nodes are selected. This method also gives the user the ability to separately control the number of nodes and the range from which these nodes are drawn. This provides the ability to adjust the range from which the nodes are selected without changing the number of neighbors that are selected. Conversely, this method also provides the ability to alter the number of neighbors that are selected while keeping the range from which they are selected fixed.

The behavior of the $LocalRand$ method differs from the k-closest method in a number of significant ways. One major difference is that this method will avoid making duplicate edge connections in undirected graphs. With the k-closest method, there is a high probability that a node's neighbors will also have the node as one of their own neighbors. This results in a total number of edge connection attempts that is considerably less than k per node. With a randomized method like $LocalRand$ this will happen less often, as the pool of potential neighbors to a given node is larger (k' vs. k), and the total number of unique connection attempts will be greater. For more details about this see [38].

Another difference is that this method will be attempting to connect longer edges than the k-closest method with the same k value. It has been previously shown that edge per edge, longer edges contribute more to roadmap quality than shorter edges [7]. At the same time, it also possible that these longer edges will be more difficult to connect, and that the advantage of having having longer edges will be offset by the fact that fewer edges are generated.

Another difference is that, as k' increases, there is a greater likelihood that the set of candidates will include nodes from different connected components. A method like k-closest that selects only the nearest node is likely to select many nodes from the same connected component. Making multiple connections with the same connected component will not contribute to the connectivity of the roadmap and will result in redundant edges. In contrast, a method like $LocalRand$ that selects some further nodes would be more likely to select nodes from different connected components. If these connections are successful they could merge these components and increase the overall connectivity of the roadmap.

In addition, there are some environments where considering only the closest neighbors can be problematic. This will particularly be a problem for environments with thin walls where there are regions of C-free that are close to each other but separated by C-obstacle (and unconnectable).

With the $LocalRand(k, k')$ method, the choice of k' will dramatically affect the resulting roadmap. As k' approaches N (nodes in roadmap), $LocalRand$ behaves like k-random. As k' approaches k, $LocalRand$ behaves more like k-closest.

Because k-closest and k-random both produce roadmaps with beneficial qualities – k-closest provides better connectivity and more roadmap edges, k-random provides a longer average edge length and shorter diameter – it is important for us to study the spectrum of roadmaps produced by $LocalRand(k, k')$ for various values of k'. This is done in section A.

C. Evaluation Metrics

It is important to compare the quality of different neighbor selection methods with a variety of performance-based metrics. The following metrics allow evaluation of (A) how effective each method is at finding connectible neighbors, (B) how good they are at producing well-connected roadmaps, and (C) the computational expense of each method.

1. Ideal Roadmap

To determine the relative quality of roadmap as the different neighbor selection parameters are changed, it is important to have an ideal case with which to normalize the results. Thus, an **ideal roadmap** will be generated for each environment. The ideal map will be connected using an all-pairs selection policy. This method, while computationally expensive, will produce the best-possible connected roadmap for the environment, because all elements in V_c will be considered as a neighbor – the other selection policies only return a subset of V_c.

2. Connectivity Metrics

The following connectivity metrics evaluate how good the methods are at finding connectible neighbors and how successful they are at producing connected roadmaps. This will help to determine how the methods affect roadmap quality.

- **Number of Edges**: The total count of roadmap edges. This metric will tell us how many edges the methods are able to produce. In general, roadmaps with more edges tend to be better connected and have a shorter diameter.
- **Local Planner Success(%)**: The percentage of successful local planner connections. This metric will tell us how effective each method is at finding connectible neighbors. While this metric trends with the number of edges generated, the exact metric differs in that the total number of connections attempted changes between methods.
- **Connectivity**: For a roadmap $R = (V, E)$, the connectivity ($conn$) is the number of pairs of nodes (p, q) for which there is a path from $p \rightarrow q$ in R. This metric indicates

how connected a roadmap is. The $conn$ of the roadmap is normalized by dividing it by the $conn$ of the ideal map. $conn = 1$ indicates that a roadmap captures the connectivity of the ideal map. $conn < 1$ indicates that there are nodes that are connected in the ideal map but not in the roadmap.

- **Diameter of Largest Connected Component**: The diameter is the length of the longest shortest path in a graph. This is important because it tells us about the structure of the roadmap and indicates how long paths in the roadmap will be. All diameters in this study are computed using the Euclidean distance metric.

3. Cost Metric

This metric shows the computational expense of each methods. This is important in determining the tradeoffs associated with the connection methods (for example, is a 5% gain in connectivity worth an order of magnitude increase in computation?).

- **CD-Calls**: The number of local-planner collision detection (CD) calls made during the **connection** phase of roadmap construction. CD-calls are a metric used to measure roadmap construction time that does not take into account computational resources.

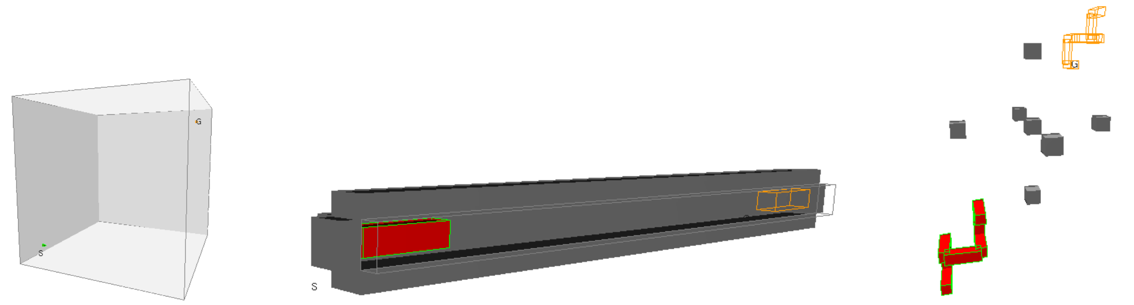
CHAPTER IV

EXPERIMENTAL STUDY

This chapter describes the set of experiments run to compare the selection policies detailed in chapter III. The motion planning problem environments studied are described, followed by other controlled parameters (node generation strategy, distance metric, etc.).

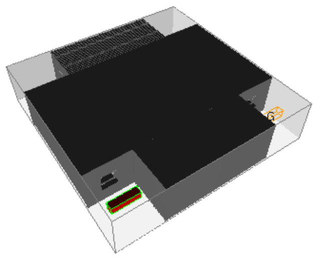
A. Environments

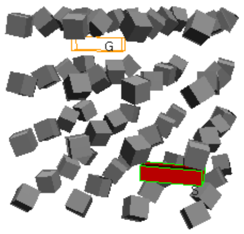
To properly study the effects of differing connection strategies, it is important to choose problem environments that are tuned to maximize the information gained through roadmap metrics. The first set shown in Fig. 1 are simple, homogeneous and serve as representatives of environments in the second set. Environments in the second set (Fig. 2, 3, 4) are more complex but share similar characteristics with environments in the first set. A general overview of all environments is discussed next.



(a) Free (rigid body) (b) Tunnel (rigid body) (c) Semi-Cluttered (linkage)

Figure 1. Baseline environments


(a) Elbow-tunnel (6 DOF)



(b) Cluttered (6 DOF)

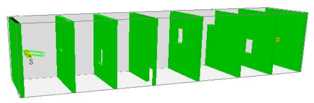

(c) Walls (7 DOF)

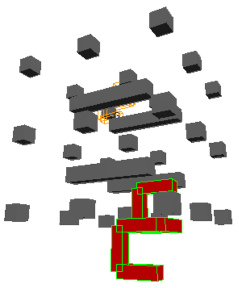
Figure 2. Low-DOF Environments

- **Free (rigid)**: [**free**] A uniformly $C\text{-}free$ environment (10x10x10), with a rigid-body cube robot (1x1x1) (Fig. 1a). This problem will help us find the k' that produces an optimal number of edges when local-planner success is guaranteed.
- **Tunnel (rigid)**: [**tu-E, tu-M, tu-H**] A homogeneous narrow-passage environment (1x1x20), with a rigid-body rectangular robot (Fig. 1b). Three permutations of this environment are studied, with different levels of difficulty (*tu-E*, *tu-M*, *tu-H*). In the easy permutation (*tu-E*), the robot is small enough to freely rotate 360 degrees in all rotational DOF. The harder permutations of the environment increase the robot size so that only 1 rotational DOF (*tu-M*) and none of the rotational DOF (*tu-H*) can rotate 360 degrees. In *tu-H*, the robot is large enough so that only translational movements are possible.
- **Semi-Cluttered (articulated, 12 DOF)**: [**freeAL**] A free environment ($10u^3$) with seven small obstacles placed uniformly in the environment (Fig. 1c). The robot is a 12-DOF articulated linkage with 6 revolute joints at various orthogonal orientations.
- **Elbow Tunnel (rigid)**: [**et-E, et-M, et-H**] An extension of the tunnel environment, this environment (Fig. 2a) consists of three narrow passages (1x1x10) connected by two "elbow" sections of free space (4x1x4). As with the baseline tunnel problem, three

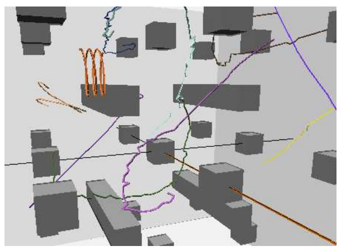
permutations of this environment are studied, with different levels of difficulty caused by proportionally increasing the robot dimensions. The same rotational restrictions are produced in these environments to match the corresponding baseline environment (*tu-E* $\rightarrow$ *et-E*, etc.)

- **Cluttered (rigid)**: [**cl-E, cl-M, cl-H**] A homogeneous cluttered environment (7x7x7, Fig. 2b) with the same rigid-body rectangular robot used in the Elbow Tunnel. Obstacles are aligned in a grid with a random orientation, with an average width of 1 unit between obstacles. Three permutations of this environment are studied, with different levels of difficulty (*cl-E, cl-M, cl-H*).
- **Walls (articulated, 7 DOF)**: [**walls**] This environment (Fig. 2c) contains a set of free areas separated by thin walls with a small opening, which the robot (2 links connected by a revolute joint) can only traverse by rotating the joint angle. The thin walls and opening will produce situations where nearest-neighbor configurations will be unconnectable, highlighting the advantages of a connection strategy (i.e. $LocalRand$) that attempts connections randomly in the local neighborhood.
- **Boxes (articulated, 12 DOF / 64 DOF / 128 DOF)**: [**boxes**] This environment (Fig. 3) contains an area cluttered with small boxes. The robot (7-link, 12-DOF articulated linkage) must traverse from one corner to the opposing corner of the environment. The swept-volume distance metric is used for computing distance between configurations. The 12 DOF problem uses a 7-link robot with orthogonal joint angles. The 64- and 128-DOF problems use a "coil"-like robot composed of many equal-size segments.
- **Forked (fixed-base articulated linkage, 32 DOF / 128 DOF / 256 DOF)**: [**forked**] A cluttered environment (Fig. 4) with several small obstacles. The robot is a fixed-base articulated linkage with a forked end that can only sample in a two-dimensional plane. Three permutations of this environment are tested: the total robot size remains

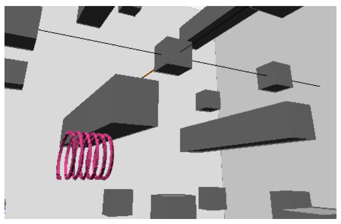
constant while the number of links increases (and the size of each link decreases).



(a) Boxes (12 DOF)

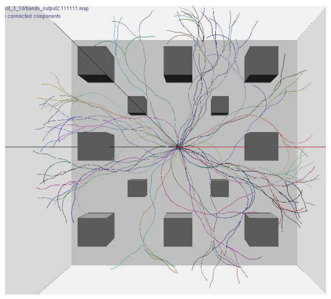


(b) Boxes (64 DOF)

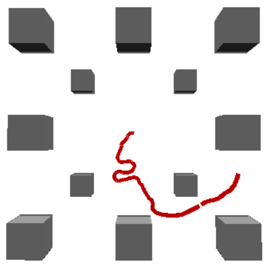


(c) Boxes (128 DOF)

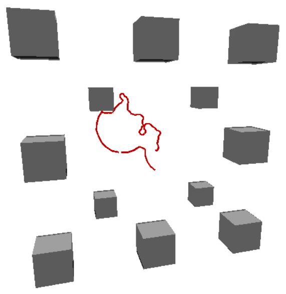
Figure 3. High-DOF Articulated Linkages (free-base)



(a) Forked (32 DOF)



(b) Forked (128 DOF)



(b) Forked (256 DOF)

Figure 4. High-DOF Articulated Linkages (fixed-base)

B. Setup

Node generation strategies and distance metric methods were selected to best fit the characteristics of each environment.

1. Node Generation

In the first set of experiments, Uniform sampling [6] was used in *free* and *semi-cluttered* environments and Obstacle-based sampling [9] in *tunnel* environments. In the second set of experiments, Uniform sampling was used in *walls*, *boxes* and *forked* environments and Obstacle-based sampling in *elbow-tunnel* and *cluttered* environments.

2. Distance Metric

Environments with a rigid-body robot used Euclidean distance, as well as those with fixed-base linkages; all free-body articulated-linkage problems used a swept distance metric. The swept distance between two configurations is the total volume the robot sweeps in the workspace when moving between configurations using the specified local planner. This distance metric is generally considered to be very accurate at the expense of being costly to compute [39].

3. Implementation and Experimental Platform

All planners were implemented using the C++ motion planning library developed by the Parasol Lab at Texas A&M University, which uses the graph from the STAPL Parallel C++ library [40]. RAPID [41] was used for collision detection computations. All computation was performed on Brazos, a major computing cluster at Texas A&M University. The processing nodes consisted of quad-core Intel Xeon processors running at 2.5 Ghz, with 15 GB of available RAM.

CHAPTER V

RESULTS

This chapter discusses the results of our study in two parts. We first study the impact of the k' parameter on the quality of roadmaps connected using the $LocalRand(k, k')$ connection method, followed by recommendations for selecting optimal k' values in different categories of motion planning problems. We then directly compare the $LocalRand(k, k')$ method with these "optimal" k' against common connection strategies employed in PRM: k-closest and k-random.

A. Selecting Parameters For LocalRand

Given the high degree of configurability of the $LocalRand(k, k')$ method, we seek in this study to a select the k' (for each k), that produces a roadmap of the highest quality. In the process we show that the $LocalRand$ method is capable of generating roadmaps that have a higher connectivity than either k-closest or k-random. We also evaluate the tradeoffs in cost and connectivity that come from using the $LocalRand$ method.

We hypothesize that for an environment with certain characteristics (e.g. narrow-passage, free, cluttered) and a robot of a certain type (e.g. high-DOF articulated linkage, rigid body) there exists an "optimal" k', or range of k', that provides a good combination of the benefits of k-random and k-closest. We seek to obtain values of k' for several k in a variety of controlled, homogeneous environments. Once these k' are fixed, this "optimal" version of $LocalRand(k, k')$ will be labeled $LR\text{-}Opt(k)$.

In order to determine this optimal set, we performed an exhaustive set of experiments over the entire range of possible k' values. In this part of our study we used 4 k values (4, 8, 16, 32). These values were selected because they were representative of the range of k values commonly used in motion planning problems. For these values of k, we connected roadmaps using $k' = k+1, k+2, ..., 128$. The results of this study were averaged over 5 runs using unique random number seeds for configuration sampling.

As environments can have wildly different underlying C-space topologies, we used the $LR\text{-}Opt(k)$ obtained from a baseline problem as the representative $LocalRand$ method for more complex environments sharing similar characteristics. For example, the $LR\text{-}Opt(k)$ obtained for a baseline narrow-passage environment will be used in a complex environment that has predominantly narrow-passage characteristics.

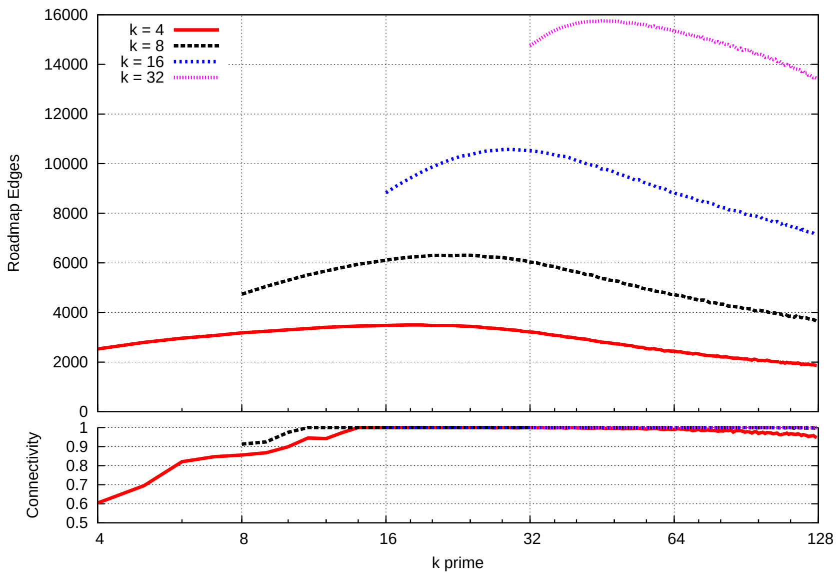


Figure 5. $LocalRand(k, k')$ – *numedges* – Tunnel (M) (6 DOF)

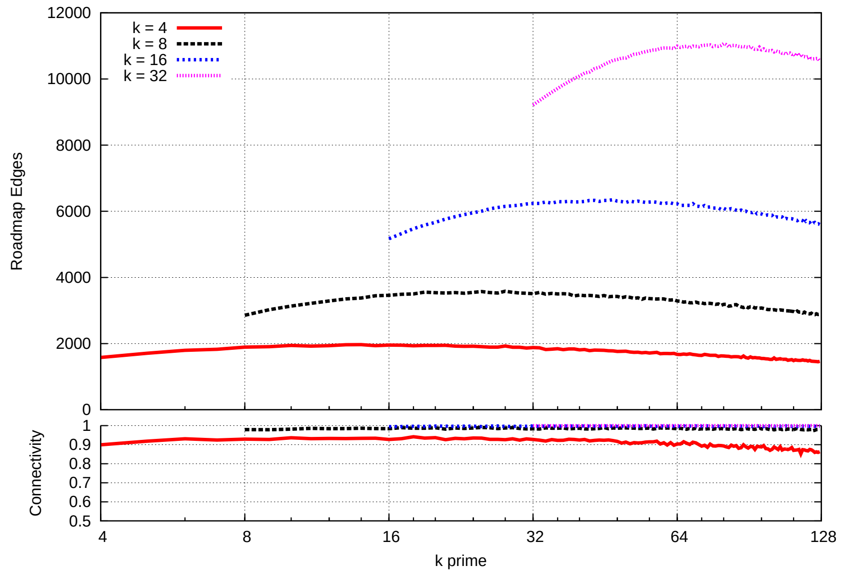


Figure 6. $LocalRand(k, k')$ – *numedges* – Semi-cluttered (12 DOF)

From the fine-grained $LocalRand(k, k')$ study, we are able to see how roadmap metrics are affected as k' increases from k. Primarily, three metrics are interesting: the number of edges connected (*numedges*), the total work performed in connection (local planner collision detection calls, or *lpcd*), and the connectivity of the roadmap (*conn*). For each k, we would like to select a k' such that $LR\text{-}Opt(k)$ maximizes *numedges* and *conn* while minimizing *lpcd*. Because the work performed in roadmap connection scales with the number of edges connected, our minimization of *lpcd* will only be used as a “tiebreaker”, after optimizing the other metrics. Figures 5 and 6 show *numedges* with *conn* for the *tu-M* and *freeAL* environments, and allow us to determine a k' that we find optimal for each k in our set.

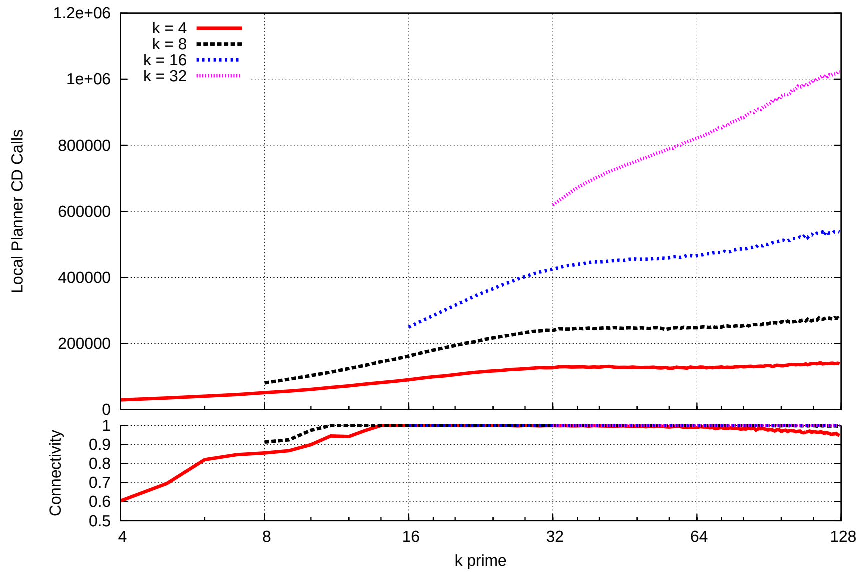


Figure 7. $LocalRand(k, k')$ – *lpcd* – Tunnel (M) (6 DOF)

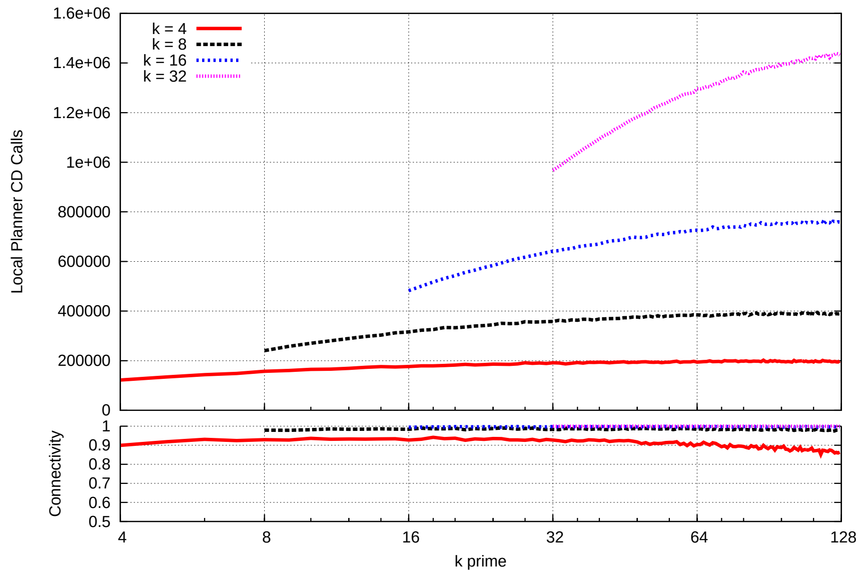


Figure 8. $LocalRand(k, k')$ – *lpcd* – Semi-cluttered (12 DOF)

B. LR-Opt(k) Selection

Figures 5 and 6 clearly show a small range of k' that result in a desirable roadmap. For small k, this k' falls between $3k$ and $4k$; for large k, between $\frac{3}{2}k$ and $3k$. As an example, consider the Tunnel (M) environment (Fig. 5). For $k = 4$, the connectivity of $LocalRand(k, k')$ is not maximized until $k' = 16$. At this point, *numedges* is nearly maximized. However, increasing k' beyond this point results in minimal edge gain, but significantly more work (*lpcd*, Fig. 7, 8). Therefore, we select $k' = 14$ for $LR\text{-}Opt(k)$ at $k = 4$.

		k'				
		free	**tu-E**	**tu-M**	**tu-H**	**freeAL**
k	**4**	12	12	16	16	12
	8	24	24	24	24	18
	16	48	48	28	28	32
	32	96	96	45	38	64

Table I. k' selected for $LR\text{-}Opt(k)$, for each k (baseline study)

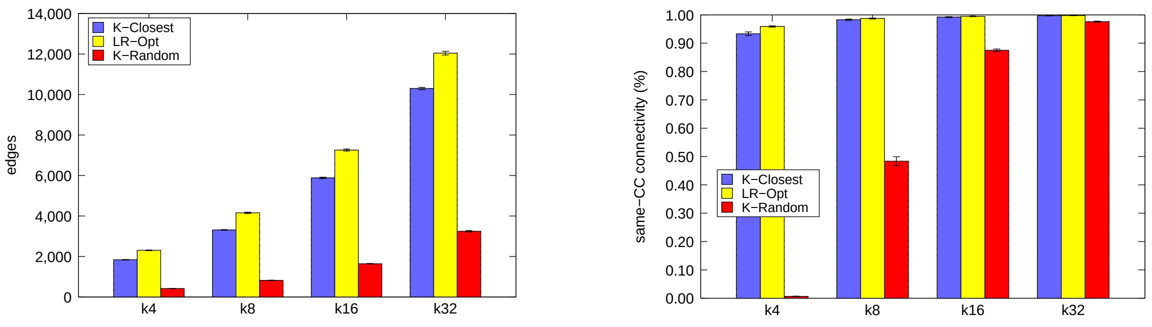
Using this methodology we select an optimal k' for each k in our set, for the environments in the baseline study (Table I). These k' are used to select the $LR\text{-}Opt(k)$ method for a given environment classification (free, narrow-passage, articulated linkage), which will be used in the remainder of this study. These selected k' range in value from 1.19 to 4 times k; environments where the connections are more difficult to achieve (*tu-M*, *tu-H*) have a smaller k', relative to k, as k increases (Table II).

		k'/k				
		free	**tu-E**	**tu-M**	**tu-H**	**freeAL**
k	**4**	3	3	4	4	3
	8	3	3	3	3	2.25
	16	3	3	1.75	1.75	2
	32	3	3	1.41	1.19	2

Table II. k' selected for $LR\text{-}Opt(k)$, shown as multiple of k

C. Comparison of Connection Methods

Next, we compare the performance of the $LocalRand$ method with the optimal k' value ($LR\text{-}Opt(k)$) to the k-closest and k-random methods. We show that the $LocalRand$ method is capable of producing better roadmaps than the k-random and k-closest methods, particularly in environments where there is more free space.



numedges *connectivity*

Figure 9. Roadmap Metrics – Elbow Tunnel (Easy)

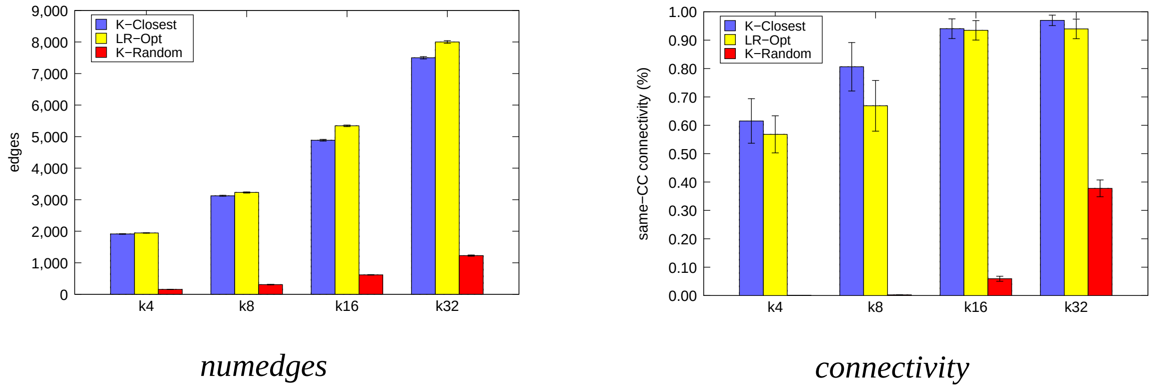


Figure 10. Roadmap Metrics – Elbow Tunnel (Medium)

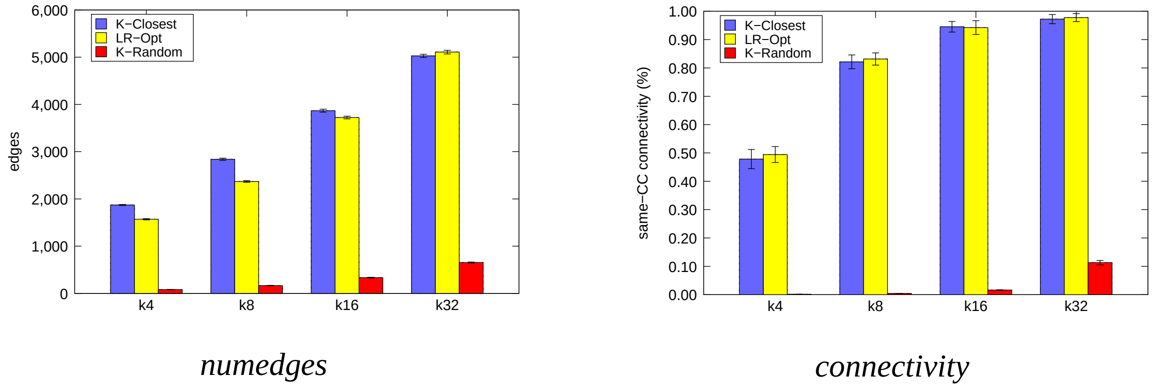


Figure 11. Roadmap Metrics – Elbow Tunnel (Hard)

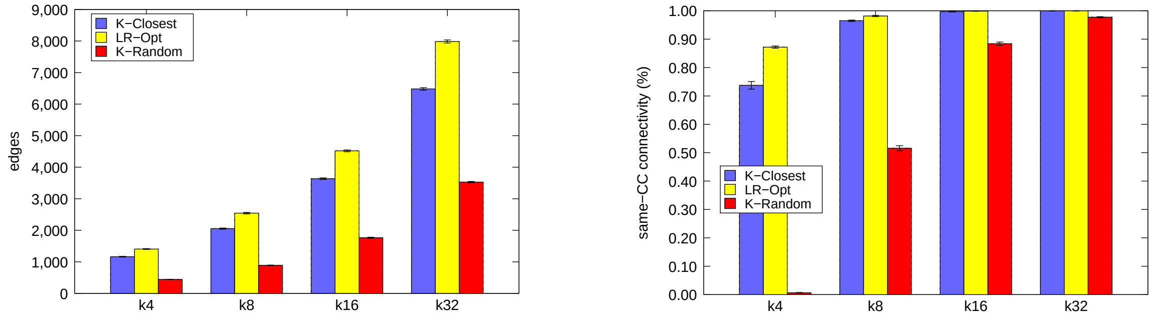


Figure 12. Roadmap Metrics – Cluttered (Easy)

To evaluate the relative performance of k-closest, k-random, and $LR\text{-}Opt(k)$, we will examine the roadmap metrics produced by these methods in each of the environments along with the number of CD-calls required by each of the methods. In this section, only the connectivity and edge count metrics will be listed (Fig. 9 through 21) as these are the metrics that most clearly show a high-quality roadmap; the remaining metrics are provided in Appendix B (Fig. 28 through 40). Recall that roadmaps with more edges and a higher connectivity are beneficial and that the number of CD-calls is indicative of the cost associated with connecting the roadmap. We are therefore looking for methods that produce roadmaps with more edges and a higher connectivity, but that do not require too many CD-calls.

A first observation is that k-closest performs relatively well compared with globally random connection (k-random). For example, consider the Cluttered (medium) environment (Fig 13, 32): more work (*lpcd*) was required to connect the roadmap than with k-closest, and both the number of edges produced (*numedges*) and connectivity (*conn*) were significantly less. As C-space becomes more complex in the medium/hard versions of this problem (Fig 11,14), and in other environments (Fig 15, 18, and 19), the detrimental effect only increases. When *lpcd* is smaller than k-closest, this is due to the fact that the longer connections attempted in k-random will fail more quickly (and thus require fewer CD calls) than shorter connections attempted by k-closest.

In comparison, applying randomness to a local neighborhood using $LR\text{-}Opt(k)$ produces, in most cases, a *higher* connectivity and a *greater* number of roadmap edges than using k-closest. This is most clearly seen in relatively free environments (Fig 9, 12, 15, and 19), where the longer average connections – we are attempting connections to neighbors that are outside of the k-closest set – are more likely to succeed.

In the *et-M* and *boxes* problems (Fig 10, 18), smaller k (4, 8) with $LR\text{-}Opt(k)$ did not provide an increase in connectivity. In fact, larger values of k (16, 32) with $LR\text{-}Opt(k)$ resulted in a decreased connectivity. This suggests that for exceptionally difficult environments (e.g. Fig. 18) – with a low local planner success rate – attempting connections beyond the closest k does not provide a benefit.

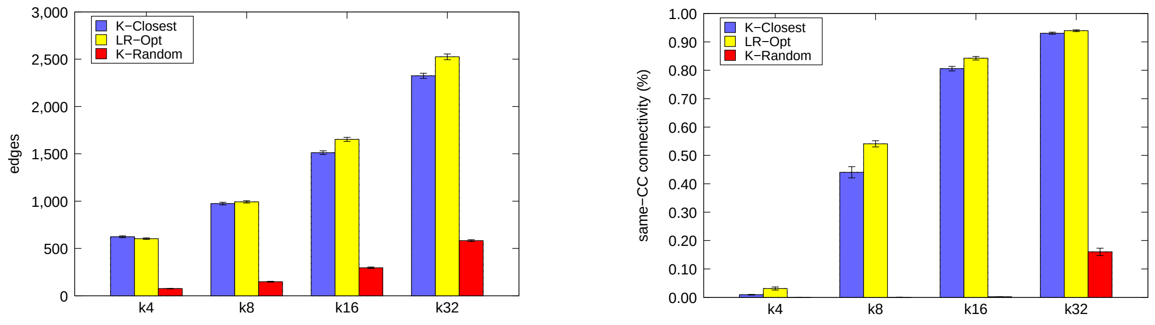


Figure 13. Roadmap Metrics – Cluttered (Medium)

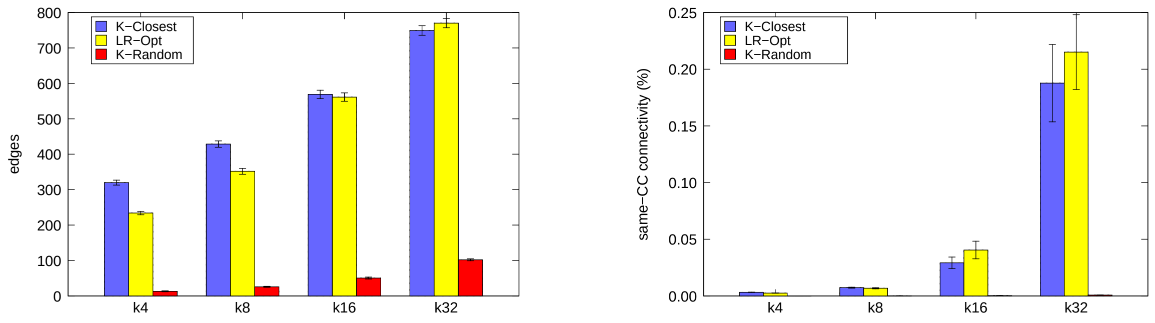


Figure 14. Roadmap Metrics – Cluttered (Hard)

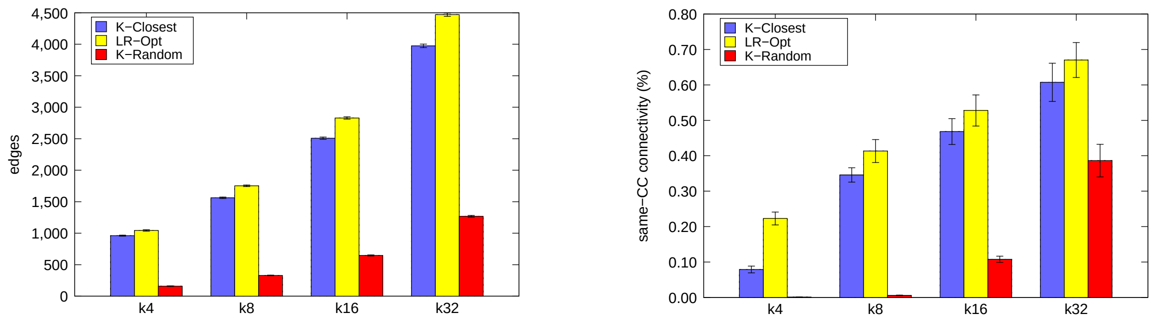


Figure 15. Roadmap Metrics – Walls (7 DOF)

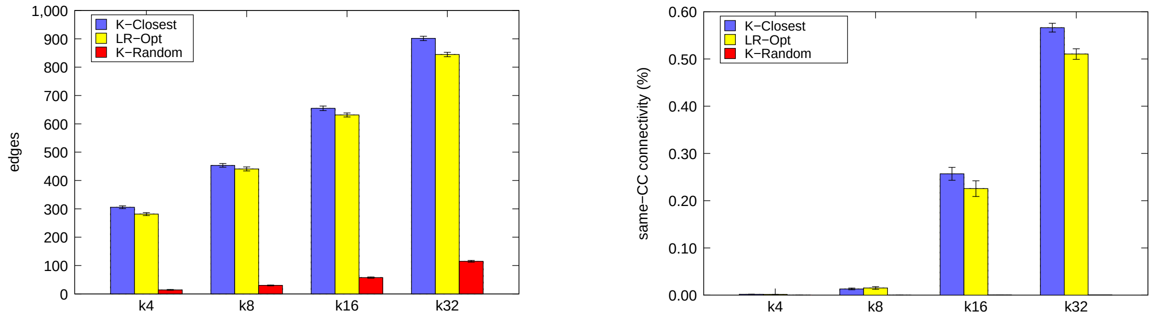


Figure 16. Roadmap Metrics – Boxes (12 DOF)

For the relatively free environments that achieve full connectivity – *et-E* (Fig. 9, 28), *cl-E* (Fig. 12, 31)) – $LR\text{-}Opt(k)$ reduces the roadmap diameter; this is desirable because roadmaps with a smaller diameter tend to produce shorter, more efficient paths.

This study examined two types of high-DOF problems: *boxes* (free-base linkage of 12, 64, and 128 DOF – Fig. 3) and *forked* (fixed-base linkage of 32, 128, and 256 DOF – Fig. 4). In all three permutations of Boxes, using $LR\text{-}Opt(k)$ resulted in a slightly decreased connectivity when compared with k-closest. However, the relative performance against k-closest remains constant as DOF increases: this suggests that the motion planning problem difficulty (local planner success rate) has a greater effect on the performance of $LR\text{-}Opt(k)$ than the DOF of the problem. In both of the fixed-base problems studied (Fig. 19, 20, 21), $LR\text{-}Opt(k)$ provided a higher connectivity than k-closest across all k studied.

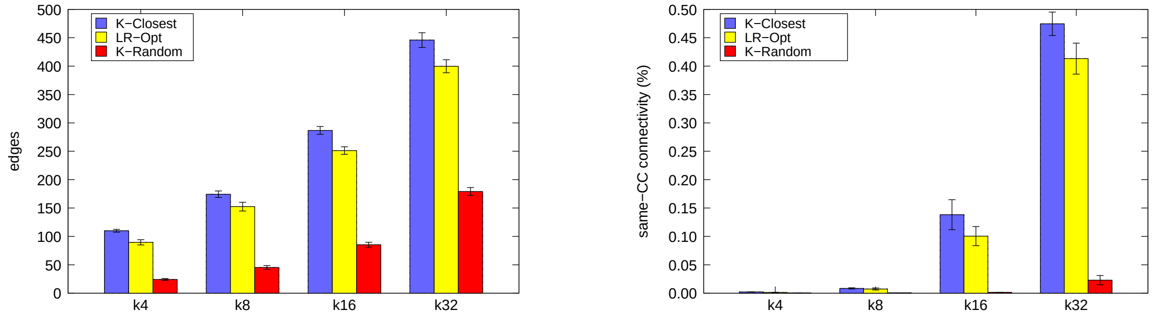


Figure 17. Roadmap Metrics – Boxes (64 DOF)

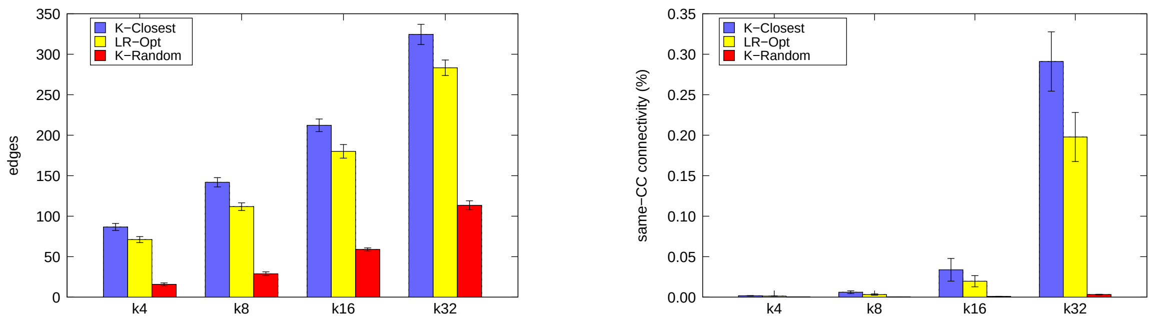


Figure 18. Roadmap Metrics – Boxes (128 DOF)

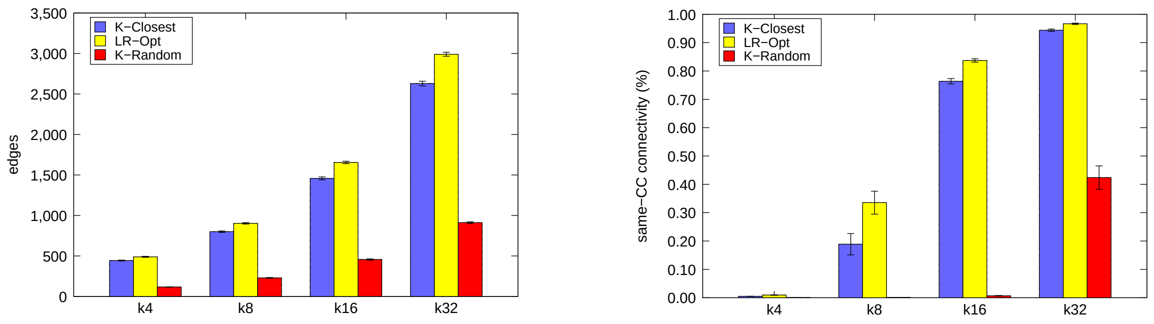


Figure 19. Roadmap Metrics – Forked (32 DOF)

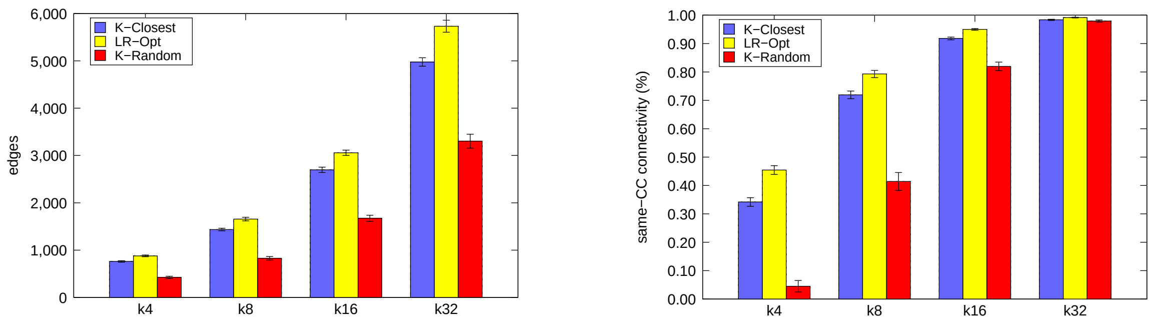
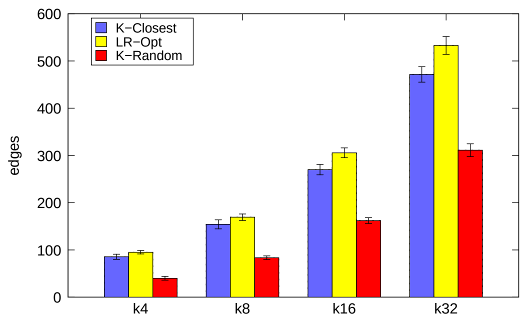


Figure 20. Roadmap Metrics – Forked (128 DOF)



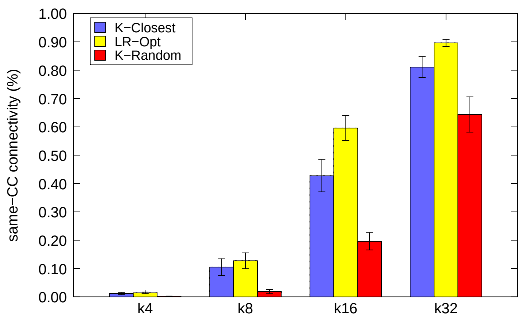


Figure 21. Roadmap Metrics – Forked (256 DOF)

CHAPTER VI

CONCLUSIONS

This study has shown that, for a variety of motion planning environments, the application of local randomization to neighbor selection can produce roadmaps of higher quality (connectivity, number of edges) and lead to a better performance than the commonly used k-closest connection method.

The baseline study, leading to the selection of $LocalRand(k, k')$, shows that a k' (size of larger set of nodes to select k-random from) exists such that $LocalRand(k, k')$ can equal or out-perform k-closest in a variety of environments. A k' of between two and three times k is recommended for difficult environments, and four times k for relatively free motion planning problems.

The scope of this study is limited to rigid-body and articulated-linkage motion planning problems, of varying degrees of complexity. As PRMs are applied to a much wider classification of motion planning problmes (e.g. simultaneous planning for multiple robots, high-DOF protein folding, closed-chain systems), future work could include studying the effectiveness of the $LocalRand(k, k')$ method into these domains. Additionally, an improved method for guiding selection of k' would aid the broad application of this strategy in practice. Also, using different k' in homogenous regions of a heterogenous problem may prove effective.

REFERENCES

[1] N. M. Amato, "Equipping CAD/CAM systems with geometric intelligence," *ACM Computing Surveys*, vol. 28, no. 4es, December 1996, Article 17, http://www.acm.org/pubs/citations/journals/surveys/1996-2 8-4es/a17-amato/.

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

[3] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*, MIT Press, Cambridge, MA, June 2005.

[4] N. M. Amato and G. Song, "Using motion planning to study protein folding pathways," *J. Comput. Biol.*, vol. 9, no. 2, pp. 149–168, 2002, Special issue of Int. Conf. Comput. Molecular Biology (RECOMB) 2001.

[5] J. H. Reif, "Complexity of the mover's problem and generalizations," in *Proc. IEEE Symp. Found. of Comp. Sci. (FOCS)*, San Juan, Puerto Rico, Oct. 1979, pp. 421–427.

[6] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, August 1996.

[7] R. Geraerts and M. H. Overmars, "Reachablility-based analysis for probabilistic roadmap planners," *J. Robot. and Autonom. Sys.*, vol. 55, pp. 824–836, 2007.

[8] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, October 1979.

[9] Y. Wu, “An obstacle-based probabilistic roadmap method for path planning,” M.S. thesis, Department of Computer Science, Texas A&M University, 1996.

[10] V. Boor, M. H. Overmars, and A. F. van der Stappen, “The Gaussian sampling strategy for probabilistic roadmap planners,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 1999, vol. 2, pp. 1018–1023.

[11] S. Wilmarth, N. M. Amato, and P. Stiller, “MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space,” Tech. Rep. TR98-022, Dept. of Computer Science, Texas A&M University, College Station, TX, Nov 1998.

[12] K. Tsianos, I. Sucan, and Lydia E. Kavraki, “Sampling-based robot motion planning: Towards realistic applications,” in *Comp. Sci. Review*, 2007, vol. 1, pp. 2–11.

[13] F. Lamiraux and D. Bonnafous, “Reactive trajectory deformation for nonholonomic systems:application to mobile robots,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2002, vol. 2, pp. 3099.–3104.

[14] J. Bruce and M. Veloso, “Real-time multi-robot motion planning with safe dynamics,” in *Multi-Robot Systems: From Swarms to Intelligent Automata*, vol. 3, pp. 159–170. 2006.

[15] K. E. Bekris and L. E. Kavraki, “Greedy but safe replanning under kinodynamic constraints,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2007.

[16] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue, “Motion planning for humanoid robots,” in *Proc. 20th Int. Symp. Robotics Research*, 2003.

[17] F. Aghili and K Parsa, “Configuration control and recalibration of a new reconfiburable robot,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2007, pp. 4077–4083.

[18] J. Cortes, T. Simeon, V. Ruiz De Angulo, D. Guieyss, M. Remaud-Simeon, and V. Tran, “A path planning approach for computing large amplitude motions of flexiable molecules,” in *Bioinformatics*, 2005, pp. 21(1):116–125.

[19] G. Sanchez and J.-C. Latombe, “On delaying collision checking in prm planning application to multi-robot coordination,” in *Int. J. Robot. Res.*, 2002.

[20] Y. Yang and O. Brock, “Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation,” in *Proc. Robotics: Sci. Sys. (RSS)*, 2006.

[21] J. Vannoy and J. Xiao, “Real-time motion planning of multiple mobile manipulators with a common task objective in shared work environments,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2007, pp. 20.–26.

[22] A. L. Olsen and H. G. Petersen, “Motion planning for gantry mounted manipulators: A ship-welding application example,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2007, pp. 4782–4786.

[23] R. Geraerts and M. Overmars, “A comparative study of probabilistic roadmap planners,” in *Proc. Int. Work. on Alg. Found. of Robot. (WAFR)*, 2002, pp. 43–57.

[24] S. M. LaValle, *Planning Algorithms*, Cambridge University Press, Cambridge, U.K., 2006, Available at http://planning.cs.uiuc.edu/.

[25] M. Branicky S. Lavalle and S. Lindemann, “On the relationship between classical grid search and probabilistic roadmaps,” *Int. J. Robot. Res.*, vol. 23, no. 7–8, pp. 673–692, 2004.

[26] L. Han D. Li, H. Yang and S. Hou, “Predicting the folding pathway of engrailed homedomain with a probablistic roadmap enhanced rection-path algorithm,” *Biophysical Journal*, vol. 94, pp. 1622–1629, 2008.

[27] R. Geraerts and M. Overmars, "Reachability-based analysis for probabilistic roadmap planners," *J. Robot. and Autonom. Sys.*, vol. 55, pp. 824–836, 2007.

[28] R. Geraerts and M. Overmars, "Sampling and node adding in probabilistic roadmap planners," *J. Robot. and Autonom. Sys.*, vol. 54, pp. 165–173, 2006.

[29] C. Nissoux, T. Simeon, and J.-P. Laumond, "Visibility based probabilistic roadmaps," in *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, 1999, pp. 1316–1321.

[30] R. Bohlin, "Path planning in practice: Lazy evaluation on a multi-resolution grid," in *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, 2001, pp. 49–54.

[31] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *CoRR*, vol. abs/1105.1186, 2011.

[32] A. Fabri, G.-J. Giezeman, L. Kettner, S. Schirra, and S. Schönherr, "On the design of CGAL a computational geometry algorithms library," *Softw. – Pract. Exp.*, vol. 30, no. 11, pp. 1167–1202, 2000.

[33] A. Yershova and S. M. LaValle, "Improving motion-planning algorithms by efficient nearest-neighbor searching," *IEEE Trans. Robot. Automat.*, vol. 23, no. 1, pp. 151–157, 2007.

[34] J. K. Uhlmann, "Satisfying general proximity/similarity queries with metric trees," in *Inf. Process. Let.*, 1991, vol. 40, pp. 175–179.

[35] T. Liu, A. W. Moore, A. Gray, and K. Yang, "An investigation of practical approximate nearest neighbor algorithms," in *Adv. in Neural Inf. Proc. Systems*, L. K. Saul, Y. Weiss, and L. Bottou, Eds., Cambridge, Massachusetts, 2005, pp. 825–832, MIT Press.

[36] E. Plaku and L.E. Kavraki, “Quantitative analysis of nearest-neighbors search in high-dimensional sampling-based motion planning,” in *Proc. Int. Work. on Alg. Found. of Robot. (WAFR)*, New York, NY, USA, July 2006.

[37] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, “An optimal algorithm for approximate nearest neighbor searching in fixed dimensions,” *Journal of the ACM*, vol. 45, no. 6, pp. 891–923, 1998.

[38] T. McMahon, S. Jacobs, B. Boyd, L. Tapia, and N. M. Amato, “Evaluation of the k-closest neighbor selection strategy for prm construction,” Technical Report, Texas A&M, College Station, TX, 2011.

[39] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo, “Choosing good distance metrics and local planners for probabilistic roadmap methods,” *IEEE Trans. Robot. Automat.*, vol. 16, no. 4, pp. 442–447, August 2000.

[40] G. Tanase, A. Buss, A. Fidel, Harshvardhan, I. Papadopoulos, O. Pearce, T. Smith, N. Thomas, X. Xu, N. Mourad, J. Vu, M. Bianco, N. M. Amato, and L. Rauchwerger, “The STAPL Parallel Container Framework,” in *Proc. ACM SIGPLAN Symp. Prin. Prac. Par. Prog. (PPoPP)*, San Antonio, TX, USA, 2011, pp. 235–246.

[41] S. Gottschalk, M. C. Lin, and D. Manocha, “OBB-tree: A hierarchical structure for rapid interference detection,” *Comput. Graph.*, vol. 30, pp. 171–180, 1996.

APPENDIX A

FIGURES - $LocalRand$ SELECTION

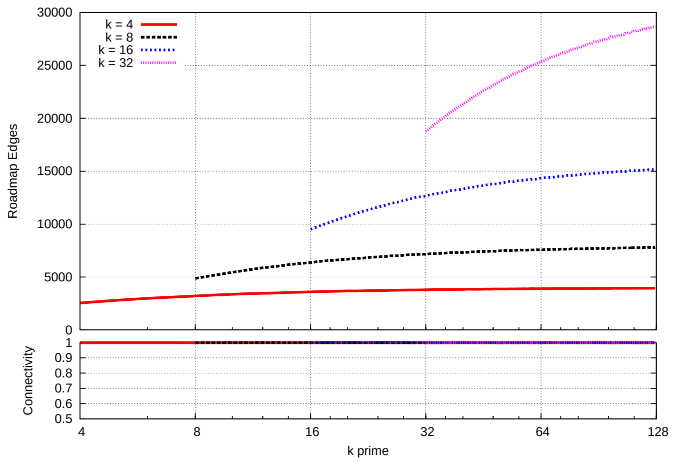


Figure 22. $LocalRand(k, k')$ – *numedges* – Free (6 DOF)

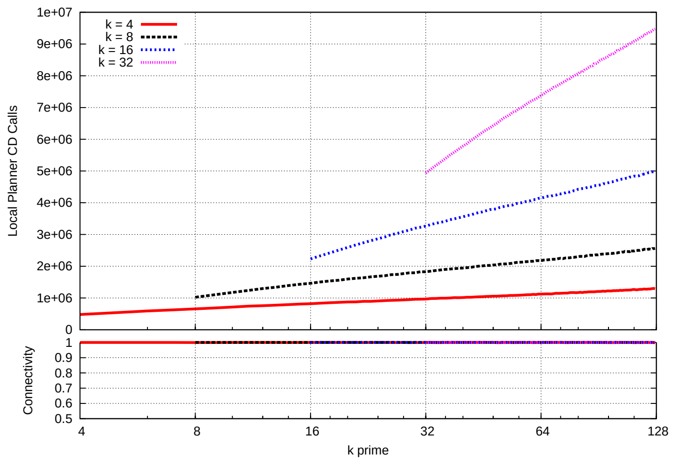


Figure 23. $LocalRand(k, k')$ – *lpcd* – Free (6 DOF)

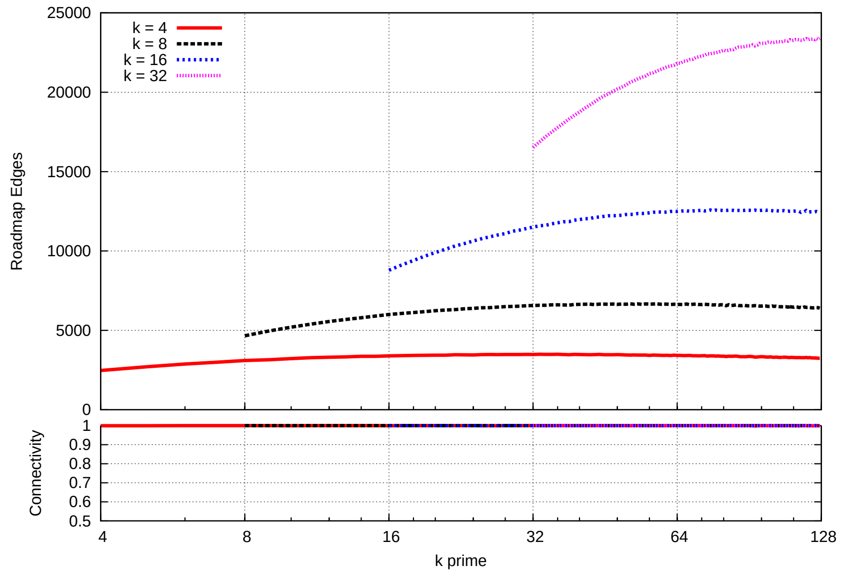


Figure 24. $LocalRand(k, k')$ – *numedges* – Tunnel (E) (6 DOF)

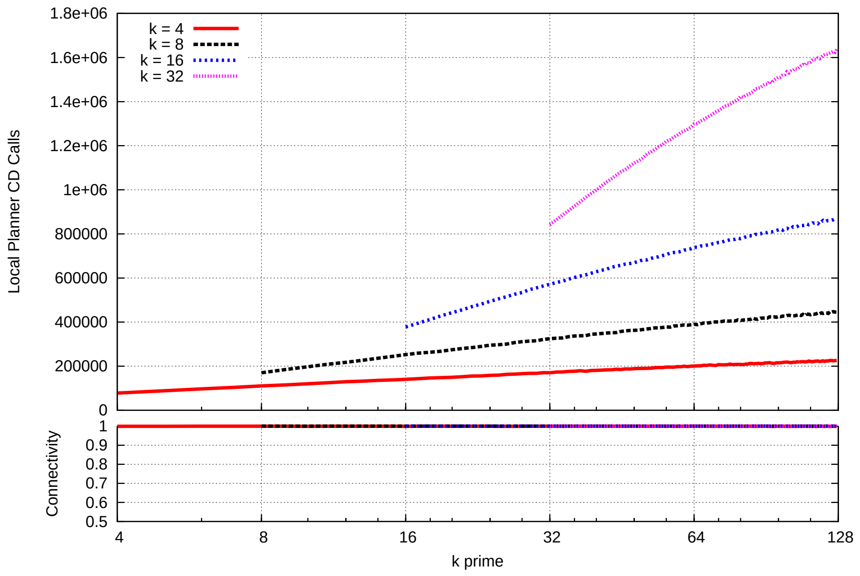


Figure 25. $LocalRand(k, k')$ – *lpcd* – Tunnel (E) (6 DOF)

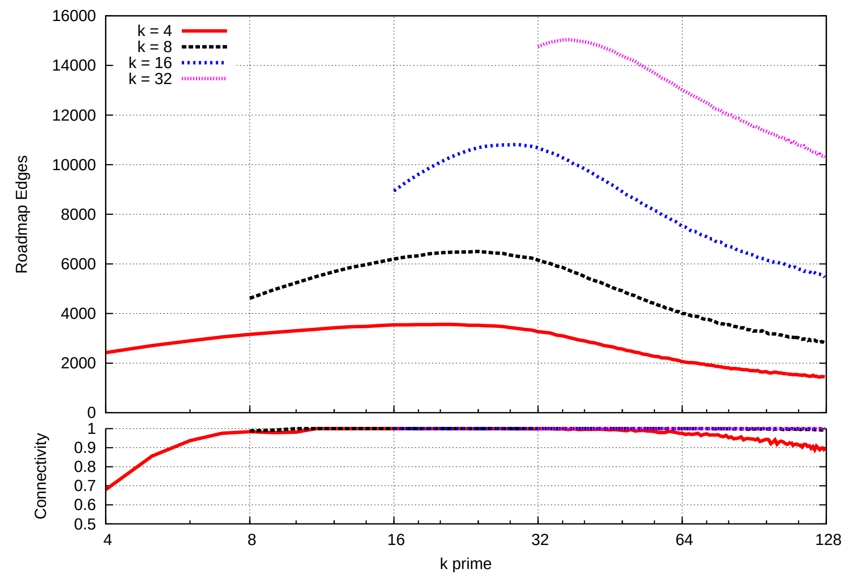


Figure 26. $LocalRand(k, k')$ – *numedges* – Tunnel (H) (6 DOF)

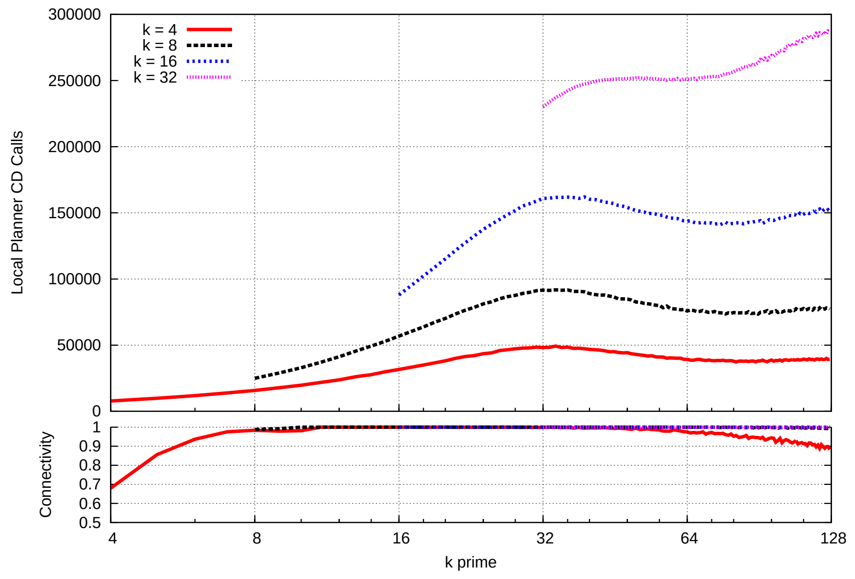


Figure 27. $LocalRand(k, k')$ – *lpcd* – Tunnel (H) (6 DOF)

APPENDIX B

FIGURES - $LocalRand$ COMPARISON

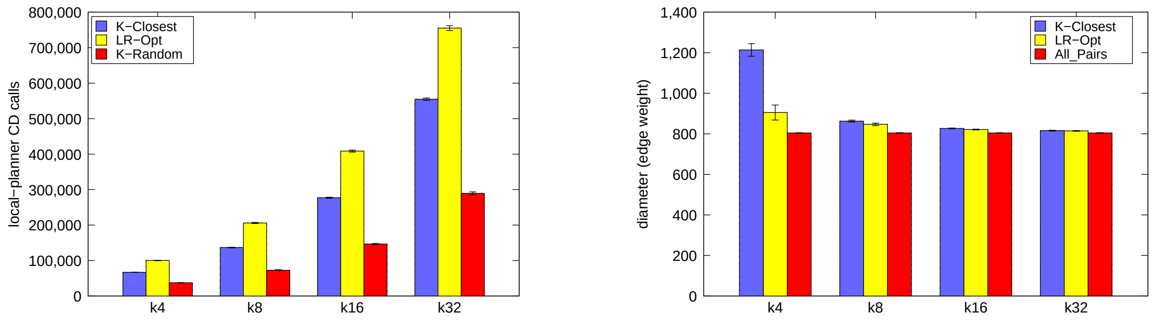


Figure 28. Additional Metrics – Elbow Tunnel (Easy)

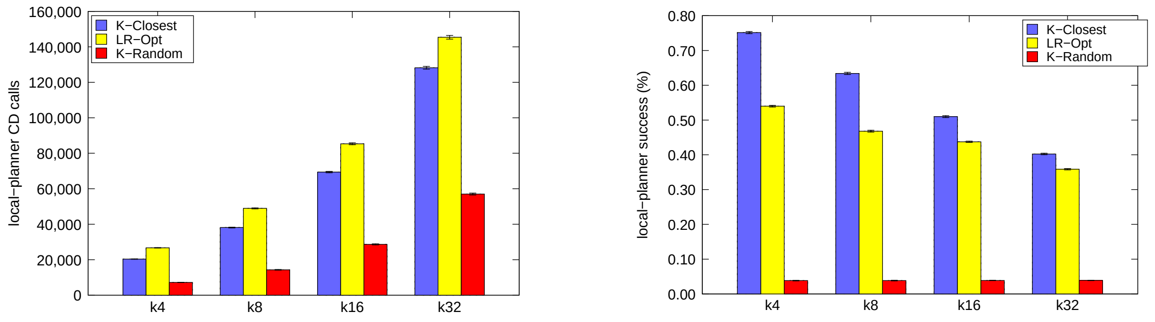


Figure 29. Additional Metrics – Elbow Tunnel (Medium)

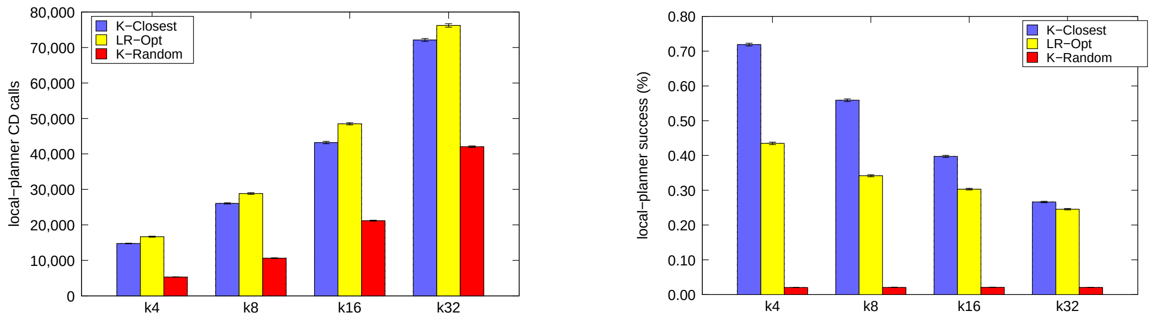


Figure 30. Additional Metrics – Elbow Tunnel (Hard)

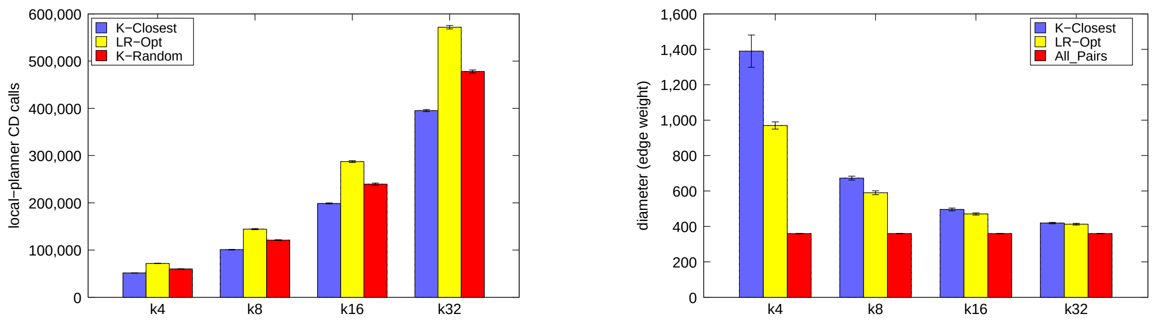


Figure 31. Additional Metrics – Cluttered (Easy)

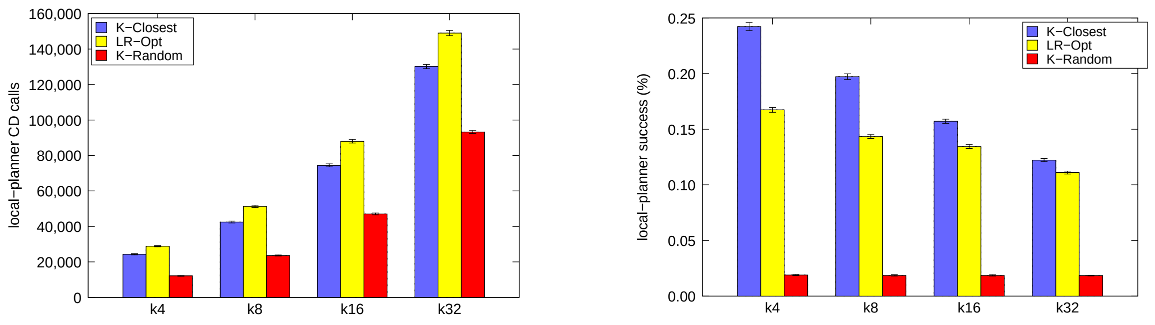


Figure 32. Additional Metrics – Cluttered (Medium)

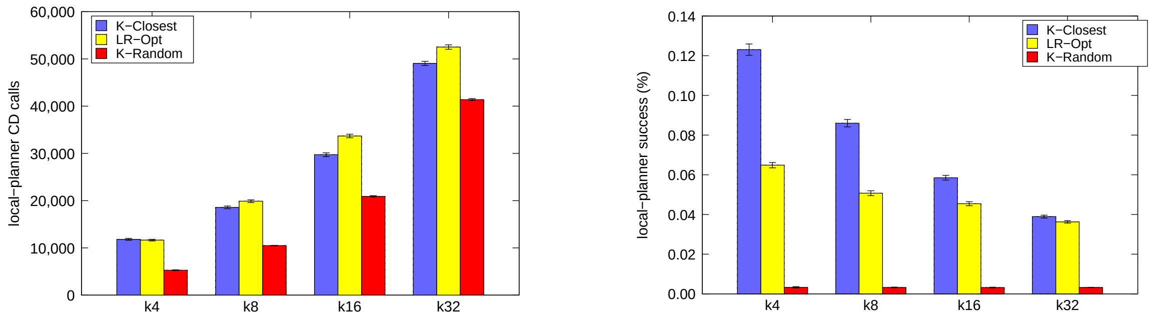


Figure 33. Additional Metrics – Cluttered (Hard)

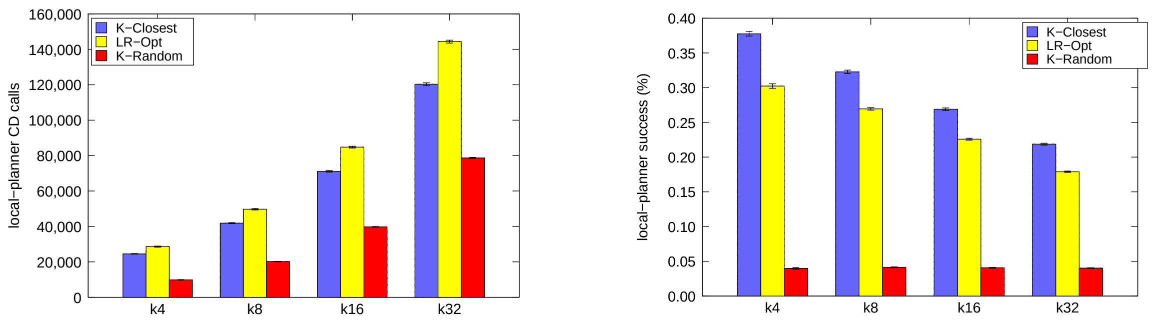


Figure 34. Additional Metrics – Walls (7 DOF)

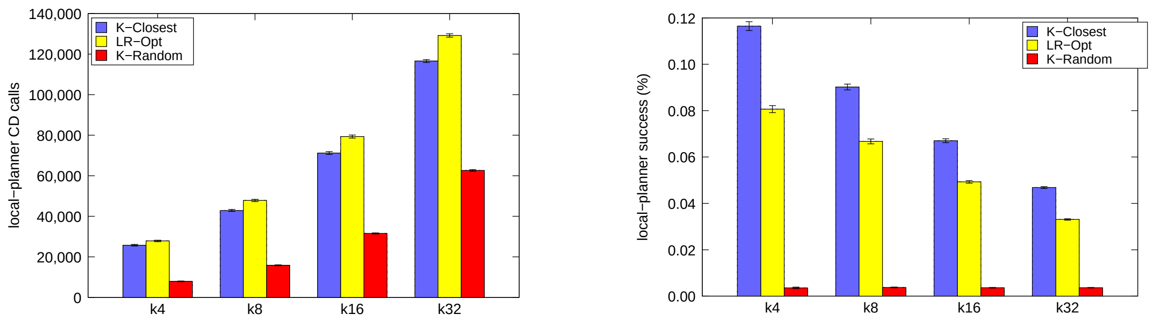


Figure 35. Additional Metrics – Boxes (12 DOF)

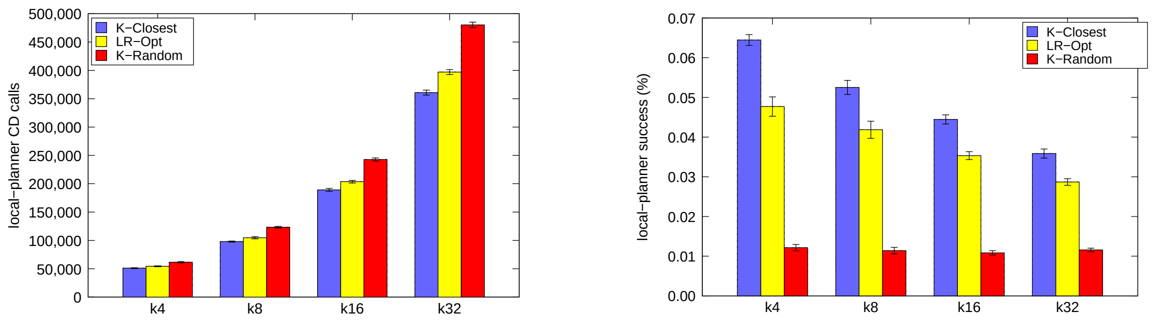


Figure 36. Additional Metrics – Boxes (64 DOF)

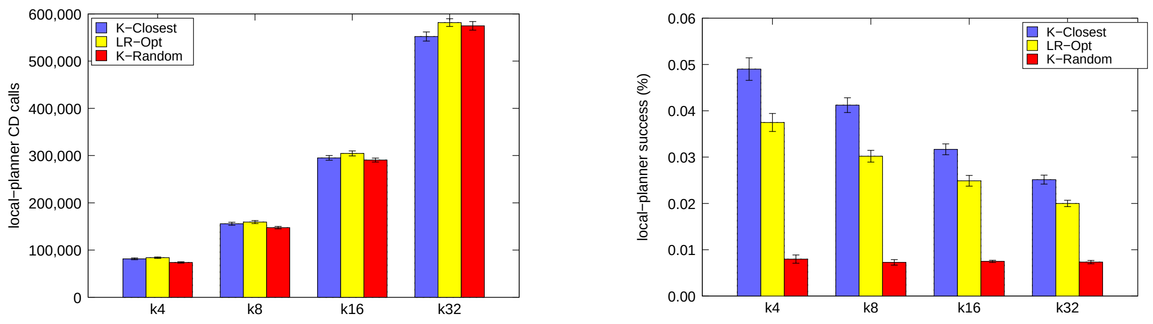


Figure 37. Additional Metrics – Boxes (128 DOF)

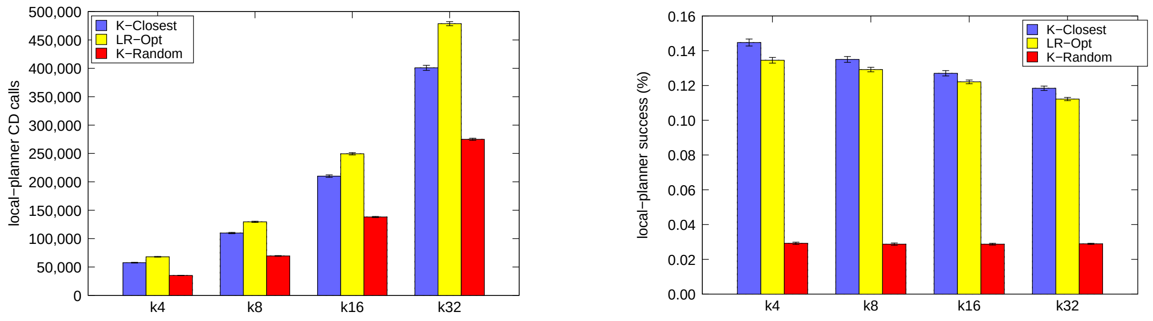


Figure 38. Additional Metrics – Forked (32 DOF)

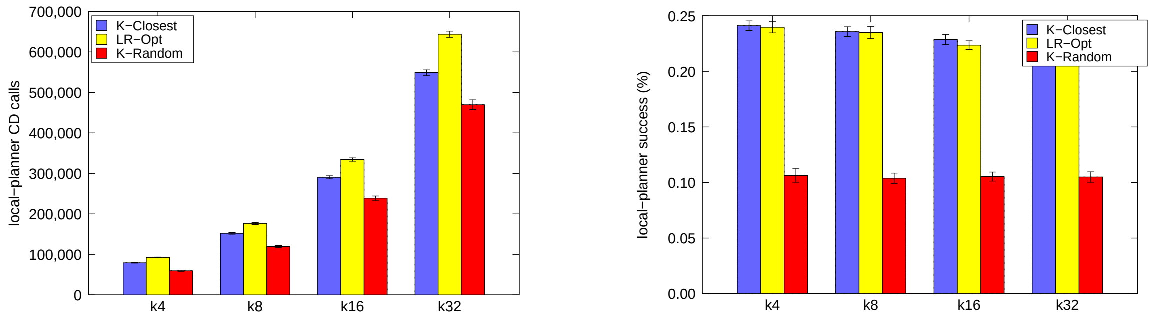
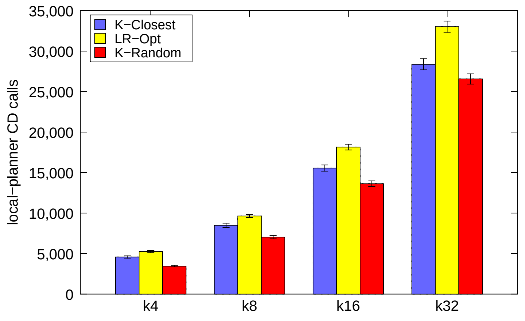


Figure 39. Additional Metrics – Forked (128 DOF)



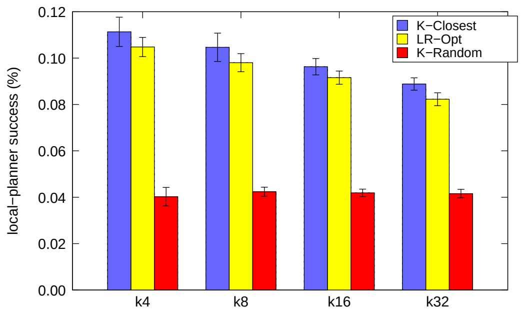


Figure 40. Additional Metrics – Forked (256 DOF)