Local Randomization in Neighbor Selection Improves PRM Roadmap Quality

Troy McMahon[1], Sam Jacobs[1], Bryan Boyd[1], Lydia Tapia[2], Nancy M. Amato[1]

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. In this paper, we propose 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. We provide a methodology for selecting the parameters k and k'. We perform an experimental comparison which shows that for both rigid and articulated robots, $LocalRand$ results in roadmaps that are better connected than the traditional k-closest policy or a purely random neighbor selection policy. The cost required to achieve these results is shown to be comparable to k-closest.**

I. Introduction

The *motion planning* problem involves finding a valid path for a movable object (robot) from a start to a goal configuration in a given environment. Motion planning is an important component of many applications, including computer-aided design [2], robotics [22], virtual reality simulations [10], and bioinformatics [4] [9]. The motion planning problem is regarded intractable, as the complexity of an exact method grows exponentially with the complexity of the robot [30].

Sampling-based methods have been able to solve many motion planning problems that exact methods cannot. One of the most common randomized methods is the Probabilistic Roadmap Method, or PRM [19]. This method randomly generates valid samples (nodes) in an environment's *configuration space* (C-space) and then attempts to connect nearby pairs of nodes using a *local planner*, a simplified planner whose objective is to generate and validate transitions between the the specified pair of nodes. The resulting graph, or *roadmap*, encodes representative feasible paths in C-space and can be queried to obtain valid paths in the environment.

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 cost of all 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 configurations 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 [10].

This research supported in part by NSF awards CRI-0551685, CCF-0833199, CCF-0830753, IIS-096053, IIS-0917266, by THECB NHARP award 000512-0097-2009, by Chevron, IBM, Intel, Oracle/Sun and by Award KUS-C1-016-04, made by King Abdullah University of Science and Technology (KAUST). Tapia supported in part by NIH Grant P20RR018754.

[1]Parasol Lab., Dept. of Comp. Sci. and Eng., Texas A&M Univ., College Station, Texas, USA. {tmcmahon,sjacobs,bcb2913,amato}@cse.tamu.edu
[2]Dept. of Comp. Sci., Univ. of New Mexico, Albequerque, New Mexico, USA. tapia@cs.unm.edu

Contribution. In this paper, we propose 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 the neighbors to be selected at random.

Intuitively, the proposed *LocalRand* method attempts to achieve the benefits associated with random sampling, while also maintaining the higher levels of local planner success inherent to selecting more local neighbors. There are differences between *LocalRand* and the traditional k-closest neighbor selection policy that 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 that k-closest. This may be beneficial as it has previously been shown that longer edges have a positive impact on roadmap quality [12].

The main contributions of this paper 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 better connected roadmaps than k-closest at a comparable cost.

Clearly, the performance of $LocalRand(k,k')$ depends crucially on the parameters k and k'. For a given k and n, the 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, we perform an extensive study over the range of possible k and k' values in some basic environments. From this study, we identify a set of optimal k' values for which the *LocalRand* method is capable of outperforming both the k-closest and k-random methods in those environments. We next consider more complex environments, and use the (k',k) values determined for the best matching basic environment. We 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 and free base articulated linkages of varying size and dimensionality.

II. Preliminaries and Related work

Identification of local transition candidates is a key step in the success of a PRM strategy. In this section we describe the role of various candidate selection strategies in PRMs and also explore how they have been used previously to achieve specific results.

A. Probabilistic Roadmap Methods

PRMs [19] are a class of sampling-based motion planners that build a graph (roadmap) in which vertices are valid robot configurations and edges represent feasible transitions between configurations. This graph then encodes representative, feasible pathways that can be used to connect given start and goal configurations.

PRMs have been applied to a wide range of problems [33]. They have been used for path planning with mobile robots [21] [8] [5], humanoid robots [20] and reconfigurable robots [1]. They have been applied to biological problems including analysis of biological structures [11] and protein folding [4]. They have been used in industrial automation and path planning with robotic manipulators [31] [35] [17] [28].

In [12], 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 [13], Geraerts and Overmars show experimentally that the main difficulty in PRM construction is constructing a roadmap whose connectivity represents the connectivity of C-space.

B. Candidate Selection Approaches

There have been a variety of proposed methods for identifying candidates for local transitions during the connection phase of PRM construction [23] The most common strategy is the so-called *k-closest*, which selects the set of k nodes that are nearest to 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 [19], OBPRM [34], and GaussPRM [7]. 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 [23].

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. The original PRM implementation included a variation of the distance method with an upper bound on the number of neighbors [19]. The distance method is commonly used for grid-based PRM methods [24]. It is also commonly used in biophysical simulations where certain cutoffs are standard, e.g., a RMSD of atom positions between molecular conformations [25]. Geraerts and Overmars [15], [14] 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 fairly well suited for generating connected roadmaps.

In [13], 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 node as in the variant, Visibility Roadmaps [27].

Geraerts and Overmars[13] also explore a *Component-based* selection strategy. This strategy attempts k connections to each connected component [23]. Depending on the number of components and the value of k, there could be a large number of attempts.

In [6], 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 points 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 [18], 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 grows.

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. In [29] Plaku et. al. study how approximate neighborhood finders perform when applied to the motion planning problem.

III. Candidate Selection Policies

In this paper we compare several strategies for selecting candidate sets for the PRM connection phase. In addition to the traditionally used k-closest strategy, we consider an *AllPairs* connection strategy (used as a baseline comparison for each method), a random connection strategy, 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 candidates based on distance calculation (i.e., k-closest and *LocalRand*) are affected by the distance metric.

- *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 it 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 chosen without changing the number of neighbors selected. Conversely, this method also permits altering the number of neighbors 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 make fewer 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 many edge connections being attempted twice, and hence the total number of unique edges will be considerably less than k per node. With a randomized method like *LocalRand* this will happen less often, and the total number of unique edges attempted will be greater. For more details see [26].

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 [12]. 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 there is more chance 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 would 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 that cannot be connected

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 VI-A.

IV. Evaluation Metrics

We assess the neighbor selection methods using a set of evaluation metrics. We selected metrics that would allow us to evaluate (A) how effective each method is at finding connectible neighbors, (B) how good it is at producing well-connected roadmaps, and (C) the computational expense of each method.

A. *AllPairs* Roadmap

The *AllPairs* roadmap captures the best possible connectivity for a node set. Because of its impracticality for non-trivial problems (connection is $O(n^2)$), we use the *AllPairs* roadmap as a baseline for comparing the other methods rather than comparing it on its own merits.

B. Connectivity Metrics

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

- **Number of Edges**: The number of roadmap edges reflects 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 defines 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**: The connectivity of a roadmap $R = (N,E)$ is the number of pairs of nodes (p, q) for which there is a path from $p \rightarrow q$ in R. The connectivity metric is normalized by the connectivity of the *AllPairs* roadmap produced using the same set of nodes, and thus represents the ability of a method to match the connectivity of the *AllPairs* strategy.
- **Diameter of Largest Connected Component**: The diameter is the length of the longest shortest path in a graph. This is important because it provides detail about the structure of the roadmap and indicates how long paths in the roadmap will be. All diameters in this paper are computed using the Euclidean distance metric.

C. Cost Metric

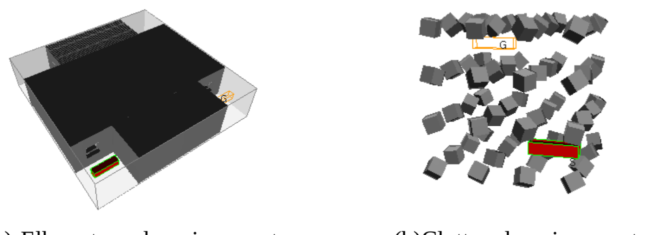
This metric shows the computational expense of each method. This is important in determining the tradeoffs associated with the connection methods. For example, is a 5% gain in connectivity worth an order of magnitude increase in computation?

- **CD-Calls**: We measure the number of collision detection (CD) calls made during the *connection* phase of roadmap construction. CD-calls are a platform-independent metric that is used to measure roadmap construction time.

V. Experimental Study

A. Environments

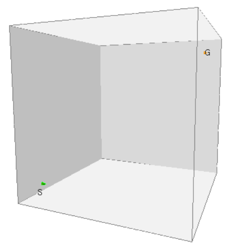
Environments used in our experiments can be categorized into two sets. The first set shown in Fig. 1 are simple, homogeneous and serve as representatives of environments in the second set. Environments in the second set (Fig. 2, 3) are more complex but share similar characteristics with environments in the first set. A general overview of all environments is discussed next.



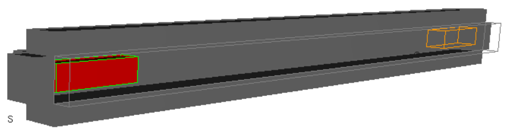
(a) Elbow-tunnel environment (b)Cluttered environment

Fig. 2: Rigid Body Environments

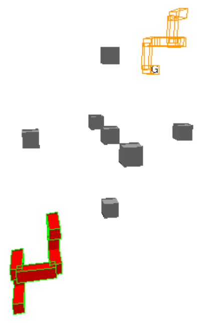
- **Free (rigid)**: [**free**] A uniformly $C\text{-}free$ environment ($10u^3$), with a rigid-body cube robot ($1u^3$) (Fig. 1(a)). This problem will help us find the k' that produces an optimal number of edges when local-planner success is guaranteed.
- **Tunnel (rigid)**: [**tu-E, tu-M, tu-H**] A homogeneous narrow-passage environment (1x1x20), with a rigid-body rectangular robot (Fig. 1(b)). 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. 1(c)). 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. 2(a)) consists of three narrow passages (1x1x10) connected by two "elbow" sections of free space (4x1x4). As with



(a) Free (rigid body)

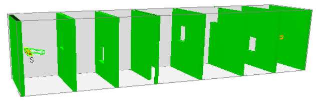


(b) Tunnel (rigid body)

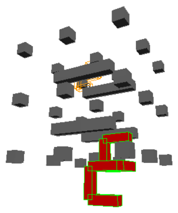


(c) Semi-Cluttered (linkage)

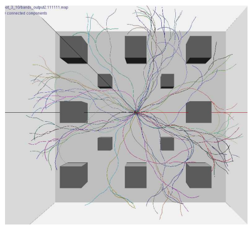
Fig. 1: Environments for baseline study



(a) Walls environment (articulated linkage, 7-DOF)



(b) Boxes environment (articulated linkage, 12-DOF)



(c) Forked environment (fixed base linkage 32-DOF)

Fig. 3: Linkage Environments

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. 2(b)) 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. 3(a)) 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)**: [**boxes**] This environment (Fig. 3(b)) 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.
- **Forked (fixed-base articulated linkage, 32-DOF)**: [**forked**] A cluttered environment (Fig. 3(c)) with several small obstacles. The robot is a fixed-base articulated linkage (32-DOF) with a forked end.

B. Setup

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

a) Node generation: In the first set of experiments, we used Uniform sampling [19] in *free* and *semi-cluttered* environments and Obstacle-based sampling [34] in *tunnel* environments. In the second set of experiments, we used Uniform sampling in *walls*, *boxes* and *forked* environments and Obstacle-based sampling in *elbow-tunnel* and *cluttered* environments.

b) Distance metric: As a distance metric, environments with a rigid-body robot used Euclidean distance; 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 [3].

c) Implementation and Experimental Platform: We implemented all planners 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 [32]. RAPID [16] 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 RAM.

VI. Results

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 a k', given 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 *LocalRand*.

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-Opt*(k).

In order to determine this set, we performed an extensive set of experiments over the 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 $k' = k+1, k+2, \ldots, 128$. 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-Opt*(k) obtained from a baseline problem as the representative *LocalRand* method for more complex environments sharing similar characteristics. For example, the *LR-Opt*(k) obtained for a baseline narrow-passage environment will be used in a complex environment that has predominantly narrow-passage characteristics.

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-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 4 and 6 show *numedges* with *conn* for the *tu-M* and *freeAL* environments, and allow us to determine a best-performing k' for each k in our set. Plots for *tu-E*, *tu-H* and *free* are not shown due to space constraints – just the chosen k'. Our full set of results for these experiments can be found in [26].

1) LR-Opt(k) Selection: The results clearly show a small range of k' values 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. 4, 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, a drop in connectivity, and significantly more work (*lpcd*). Therefore, we select $k' = 14$ for *LR-Opt*(k) at $k = 4$. This decrease in quality for k' greater than this value can be attributed to attempting longer connections, with a lower local-planner success rate.

		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-Opt*(k) for each k (baseline study)

Using this methodology we select a high-performing k' for each k in our set, for the environments in the baseline study (Table I). These k' are used to select the *LR-Opt*(k) method for a given environment classification (free, narrow-passage, articulated linkage), which will be used in the remainder of this study.

B. Comparison of connection methods

Next, we compare the performance of the *LocalRand* method with the high-performing k' value (*LR-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.

To evaluate the relative performance of k-closest, k-random, and *LR-Opt*(k), we will examine the number of edges and connectivity of roadmaps generated by these methods in each of the environments along with the number of CD-Calls required by each of the methods. These results are in Figures 7, 8, 9, 10, 11, 12, 13, 14, and 15 (error bars show a one-sided confidence interval of 95%). 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. For this set of experiments, roadmap

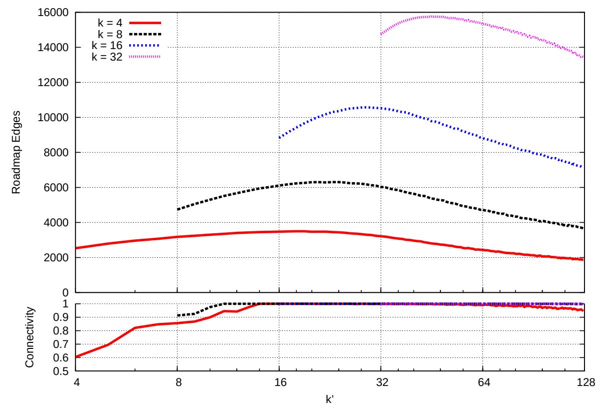


Fig. 4: *LocalRand*(k,k') – *numedges* – Tunnel (M) (6-DOF)

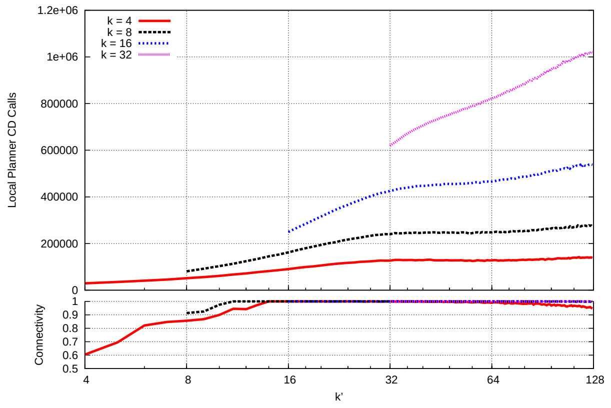


Fig. 5: *LocalRand*(k,k') – *lpcd* – Tunnel (M) (6-DOF)

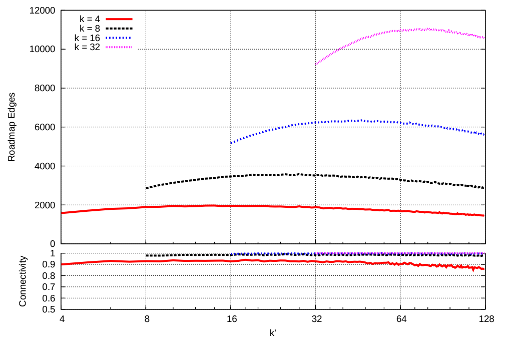


Fig. 6: *LocalRand*(k,k') – *numedges* – Semi-cluttered (12-DOF)

statistics are averaged over 20 runs in each environment using unique seeds for configuration sampling.

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 11): 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 9, 12), and in other environments (Fig 13, 14, and 15), 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 7, 10, 13, and 15), 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 8, 14), 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 (Fig. 14) – with a low local planner success rate – attempting connections beyond the closest k does not provide a benefit.

For the relatively free environments that achieve full connectivity – *et-E* (Fig. 7), *cl-E* (Fig. 10)) – $LR\text{-}Opt(k)$ reduces the roadmap diameter; this is desirable because roadmaps with a smaller diameter tend to produce shorter, more efficient paths.

VII. Conclusions

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

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

In future work, we would like to extend our study to additional types of motion planning problems (multiple robots, high-DOF protein folding, closed-chain systems).

References

[1] F. Aghili and K Parsa. Configuration control and recalibration of a new reconfiburable robot. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 4077–4083, 2007.

[2] N. M. Amato. Equipping CAD/CAM systems with geometric intelligence. *ACM Computing Surveys*, 28(4es), December 1996. Article 17, http://www.acm.org/pubs/citations/journals/surveys/1996-2 8-4es/a17-amato/.

[3] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo. Choosing good distance metrics and local planners for probabilistic roadmap methods. *IEEE Trans. Robot. Automat.*, 16(4):442–447, August 2000.

[4] N. M. Amato and G. Song. Using motion planning to study protein folding pathways. *J. Comput. Biol.*, 9(2):149–168, 2002. Special issue of Int. Conf. Comput. Molecular Biology (RECOMB) 2001.

[5] K. E. Bekris and L. E. Kavraki. Greedy but safe replanning under kinodynamic constraints. In *IEEE International Conference on Robotics and Automation*, 2007.

[6] Robert Bohlin. Path planning in practice; lazy evaluation on a multi-resolution grid. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 49–54, 2001.

[7] V. Boor, M. H. Overmars, and A. F. van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 2, pages 1018–1023, May 1999.

[8] J. Bruce and M. Veloso. Real-time multi-robot motion planning with safe dynamics. In *Multi-Robot Systems: From Swarms to Intelligent Automata*, volume 3, 2006.

[9] D. Brutlag, M.S. Apaydin, C. Guestrin, D. Hsu, C. Varma, A. Singh, and J.-C. Latombe. Using robotics to fold proteins and dock ligands. In *Bioinformatics*, page 18:S74.S83, 2002.

[10] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George A. Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, June 2005.

[11] 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*, pages 21(1):116–125, 2005.

[12] R. Geraerts and M. H. Overmars. Reachablility-based analysis for probabilistic roadmap planners. *Robotics and Autonomous Systems*, 55:824–836, 2007.

[13] Roland Geraerts and Mark Overmars. A comparative study of probabilistic roadmap planners. In *Workshop on the Algorithmic Foundations of Robotics*, pages 43–57, 2002.

[14] Roland Geraerts and Mark Overmars. Sampling and node adding in probabilistic roadmap planners. *In Journal of Robotics and Autonomous Systems*, 54:165–173, 2006.

[15] Roland Geraerts and Mark Overmars. Reachability-based analysis for probabilistic roadmap planners. *In Journal of Robotics and Autonomous Systems*, 55:824–836, 2007.

[16] S. Gottschalk, M. C. Lin, and D. Manocha. OBB-tree: A hierarchical structure for rapid interference detection. *Comput. Graph.*, 30:171–180, 1996. Proc. SIGGRAPH '96.

[17] John Xiao J. Vannoy. Real-time motion planning of multiple mobile manipulators with a common task objective in shared work environments. In *IEEE International Conference on Robotics and Automation*, pages 20.–26, 2007.

[18] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *CoRR*, abs/1105.1186, 2011.

[19] 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.*, 12(4):566–580, August 1996.

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

[21] F. Lamiraux and D. Bonnafous. Reactive trajectory deformation for nonholonomic systems:application to mobile robots. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 3099.–3104, 2002.

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

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

[24] S. M. Lavalle, M. S. Branicky, and S. R. Lindemann. On the relationship between classical grid search and probabilistic roadmaps. *Int. J. Robot. Res.*, 23(7–8):673–692, 2004.

[25] Dawei Li, Hijun Yang, Li Han, and Shuanghong Hou. Predicting the folding pathway of engrailed homedomain with a probablistic roadmap enhanced rection-path algorithm. *Biophysical Journal*, 94:1622–1629, 2008.

[26] Troy McMahon, Sam Jacobs, Bryan Boyd, Lydia Tapia, and Nancy Amato. Evaluation of the k-closest neighbor selection strategy for prm construction. Technical Report TR12-002, Texas A&M, College Station Tx., 2011.

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

[28] A. L. Olsen and H. G. Petersen. Motion planning for gantry mounted manipulators: A ship-welding application example. In *IEEE International Conference on Robotics and Automation*, pages 4782–4786, 2007.

[29] E. Plaku and L.E. Kavraki. Quantitative analysis of nearest-neighbors search in high-dimensional sampling-based motion planning. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, July 2006.

[30] J. H. Reif. Complexity of the mover's problem and generalizations. In *Proc. 1EEE Symp. Foundations of Computer Science (FOCS)*, pages 421–427, San Juan, Puerto Rico, October 1979.

[31] G. Sanchez and J.-C. Latombe. On delaying collision checking in prm planning application to multi-robot coordination. In *Int. Journal of Robotics Research*, 2002.

[32] Gabriel Tanase, Antal Buss, Adam Fidel, Harshvardhan, Ioannis Papadopoulos, Olga Pearce, Timmie Smith, Nathan Thomas, Xiabing Xu, Nedal Mourad, Jeremy Vu, Mauro Bianco, Nancy M. Amato, and Lawrence Rauchwerger. The STAPL Parallel Container Framework. In *Proc. ACM SIGPLAN Symp. Prin. Prac. Par. Prog. (PPoPP)*, pages 235–246, San Antonio, Texas, USA, 2011.

[33] Konstantinous I. Tsianos, Ioan A. Sucan, and Lydia E. Kavraki. Sampling-based robot motion planning: Towards realistic applications. In *Computer Science Review*, volume 1, pages 2–11, 2007.

[34] Y. Wu. An obstacle-based probabilistic roadmap method for path planning. Master's thesis, Department of Computer Science, Texas A&M University, 1996.

[35] Y. Yang and O. Brock. Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation. In *Robotics: Science and Systems*, 2006.

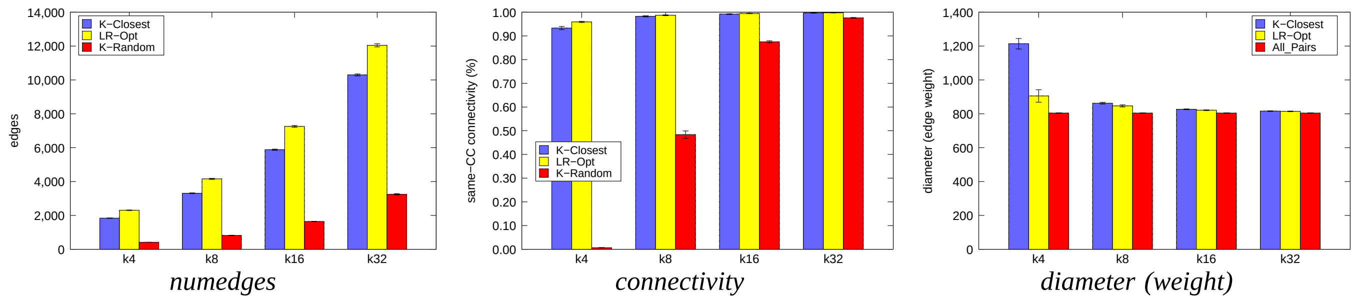


Fig. 7: Roadmap Metrics – Elbow Tunnel (Easy)

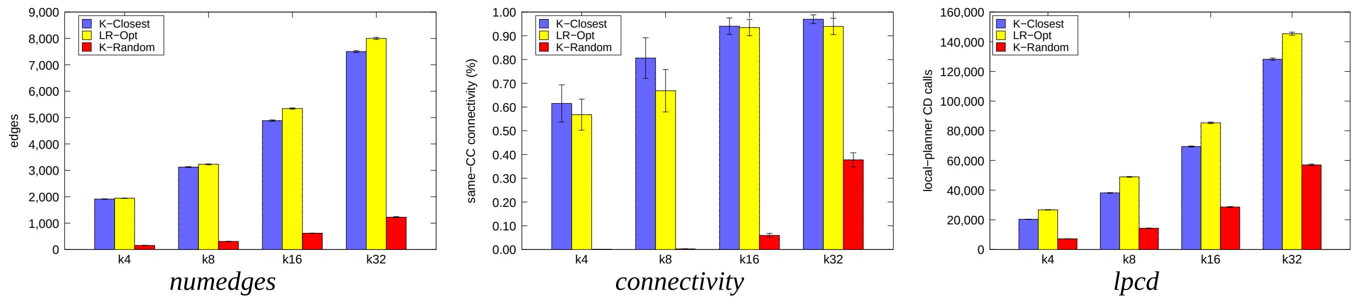


Fig. 8: Roadmap Metrics – Elbow Tunnel (Medium)

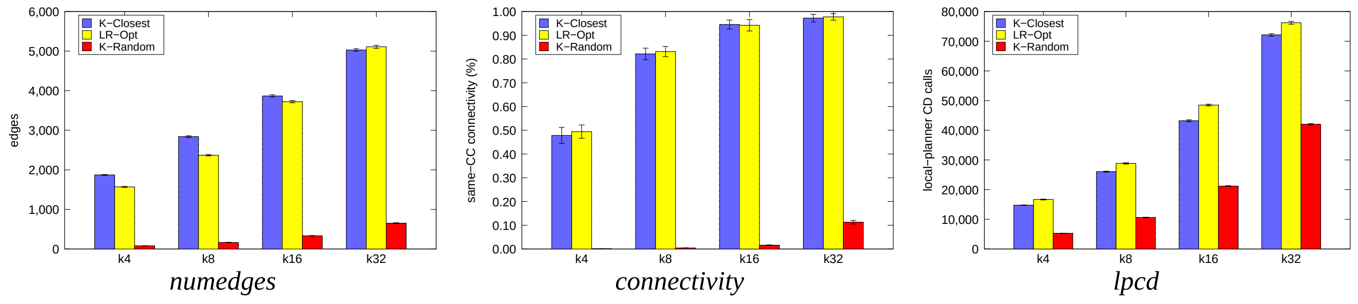


Fig. 9: Roadmap Metrics – Elbow Tunnel (Hard)

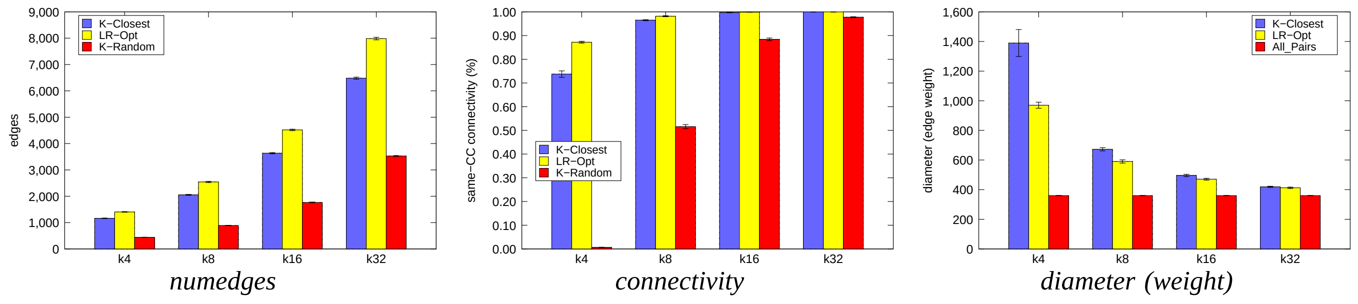


Fig. 10: Roadmap Metrics – Cluttered (Easy)

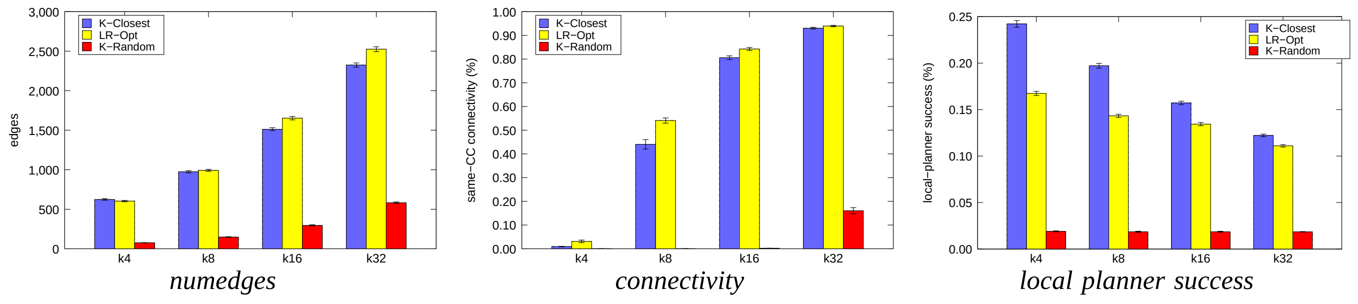


Fig. 11: Roadmap Metrics – Cluttered (Medium)

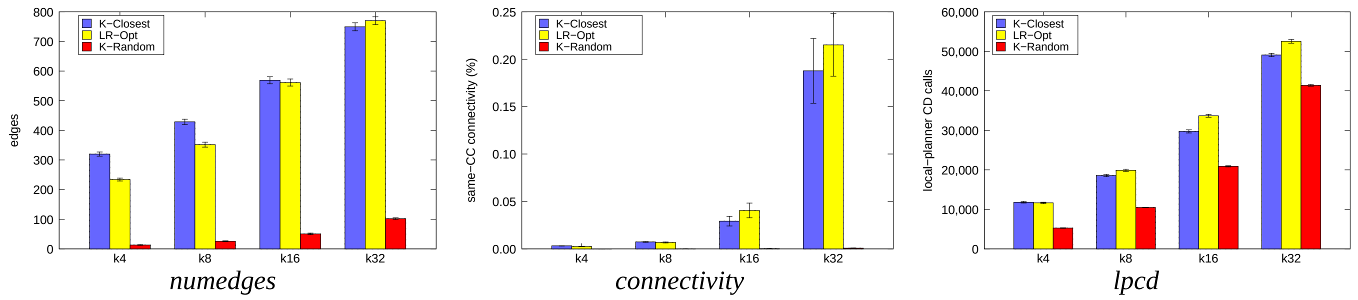


Fig. 12: Roadmap Metrics – Cluttered (Hard)

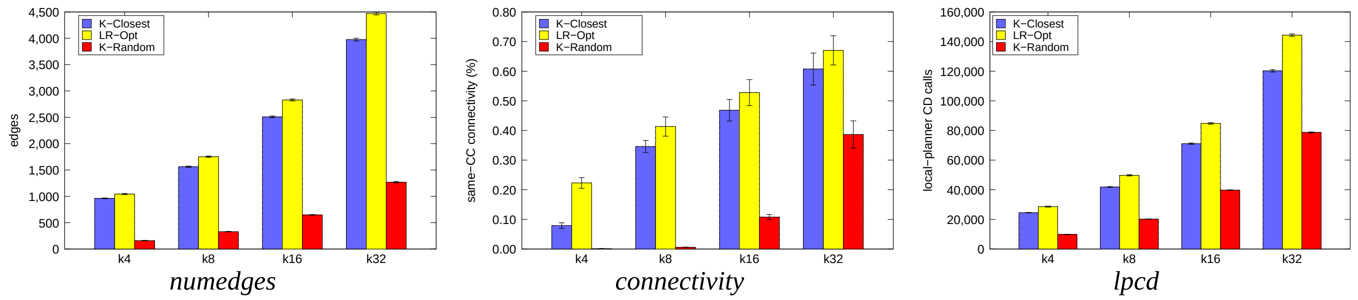


Fig. 13: Roadmap Metrics – Walls (7-DOF)

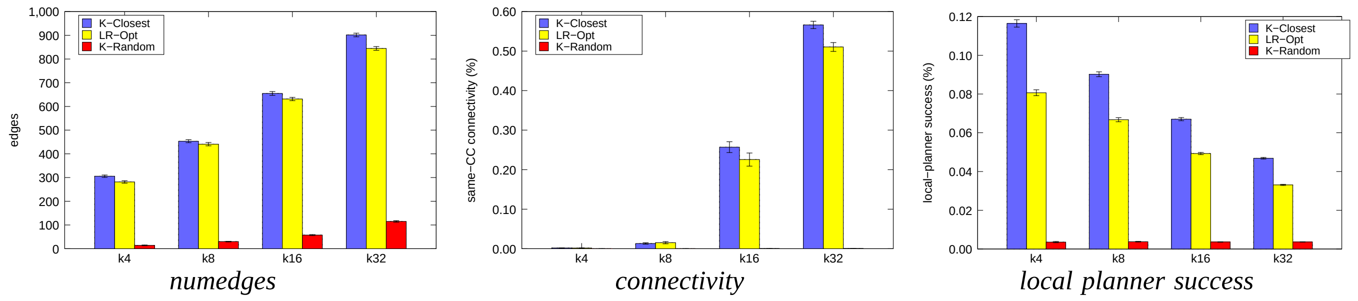


Fig. 14: Roadmap Metrics – Boxes (12-DOF)

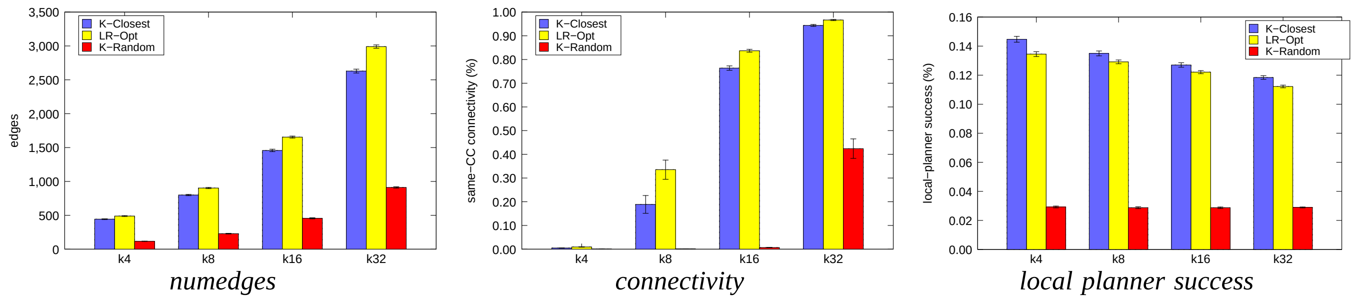


Fig. 15: Roadmap Metrics – Forked (12-DOF)