Towards Small Asymptotically Near-Optimal Roadmaps

James D. Marble and Kostas E. Bekris

***Abstract*— An exciting recent development is the definition of sampling-based motion planners which guarantee asymptotic optimality. Nevertheless, roadmaps with this property may grow too large and lead to longer query resolution times. If optimality requirements are relaxed, existing asymptotically near-optimal solutions produce sparser graphs by removing redundant edges. Even these alternatives, however, include all sampled configurations as nodes in the roadmap. This work proposes a method, which can reject redundant samples but does provide asymptotic coverage and connectivity guarantees, while keeping local path costs low. Not adding every sample can significantly reduce the size of the final roadmap. An additional advantage is that it is possible to define a reasonable stopping criterion for the approach inspired by previous methods. To achieve these objectives, the proposed method maintains a dense graph that is used for evaluating the performance of the roadmap with regards to local path costs. Experimental results show that the method indeed provides small roadmaps, allowing for shorter query resolution times. Furthermore, smoothing the final paths results in an even more advantageous comparison against alternatives with regards to path quality.**

I. Introduction

Sampling-based planning is a practical approach to solving complex and high-dimensional motion planning problems. Roadmap methods, such as the PRM [1], build up knowledge about the configuration space (C-space) by sampling and connecting configurations when possible. Traditionally, the focus has been on finding feasible solutions quickly and many methods returned paths with cost arbitrarily larger than optimal [2]. Smoothing can improve solutions and algorithms exist that produce roadmaps with paths deformable to optimal ones by reasoning about homotopic classes [3], [4]. Hybridization graphs combine multiple solutions into a higher quality one that uses the best portions of each input path [5]. These techniques, however, can be expensive for the online resolution of a query.

Alternatively, it is possible to invest more time and provide solutions which converge to optimal as more samples are added, i.e., providing asymptotic optimality guarantees [6]. Roadmaps with this property are desirable for their low solution cost but it is not clear when to stop sampling. This can result in large roadmaps, which imply significant costs during construction, storage, transmission and online query resolution. The k−PRM* algorithm [7] minimizes roadmap density while still providing asymptotic optimality. Even so, the size of roadmaps produced by k−PRM* can be high, resulting in slow online query resolution times.

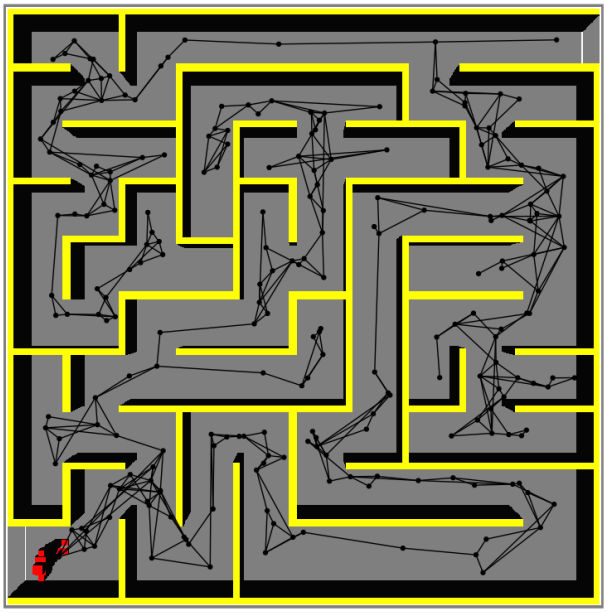

Fig. 1. A small, sparse roadmap covering the maze environment, which corresponds to an $SE(2)$ challenge. Complex areas of the space are covered by a denser part of the roadmap so that low cost local paths are maintained.

The authors have shown in previous work that a viable alternative is to compute roadmaps with asymptotically near-optimal guarantees [8], [9]. Graph spanners provide a framework for removing redundant edges from a roadmap while still maintaining paths arbitrarily close to optimum. This idea was applied in an incremental manner within the k−PRM* framework to produce the Incremental Roadmap Spanner (IRS) algorithm [9]. IRS results in sparser roadmaps than k−PRM* at the cost of a small degradation in path quality. The path quality degradation is quite smaller in practice than the theoretical guarantees of the method and is almost non-existent if smoothing is applied. Nevertheless, IRS maintains the same number of nodes as k−PRM*.

The roadmap method in this work does not add every C-space sample as a node but does provide asymptotic coverage and connectivity. It is also experimentally that it returns low path costs. As edges can be redundant for a near-optimal planner, nodes can also be redundant for coverage and connectivity. Moreover, nodes tend to impose a significant space overhead as they need to store the configuration they correspond to. Even in an asymptotically near-optimal roadmap, introducing new nodes forces the addition of more edges, which further increases the roadmap size.

This work is supported by NSF CNS 0932423. Any opinions expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsor.

The authors are with the Department of Computer Science and Engineering, University of Nevada, Reno, NV, 1664 N. Virginia St., MS 171, Reno, NV, 89557. Corresponding author: bekris@cse.unr.edu.

An advantage of IRS2 is that it gives rise to a stopping criterion. Similar to the Visibility roadmap [10], a sample that does not satisfy desirable path cost properties is a failed iteration. The number of consecutive failures is a probabilistic measure of how close the roadmap is to a solution that provides the desirable properties. In particular, M consecutive failures imply that there is a $1 - \frac{1}{M}$ probability of covering the C-space and a $\frac{1}{M}$ probability of improving the solution the following iteration. Consequently, it is possible to devise a stopping criterion given the desired number of consecutive failures, which is not straightforward for approaches that include all the samples in the roadmap.

Beyond the roadmap to be returned, IRS2 also maintains a dense graph. Every sample is added to the dense graph and edges are created connecting all of the neighbors within a δ-ball that it can connect to with a collision-free path, where δ is an input parameter specifying the maximum distance that nodes can be connected. This sample is added to the sparse roadmap if it provides coverage, connectivity or improvement in path cost. Coverage and connectivity are evaluated in a manner similar to the Visibility roadmap [10]. A sample improves the path cost of the local roadmap if there is no path in the existing roadmap that is shorter than t times a path through the candidate configuration.

Experiments on $SE(2)$ and $SE(3)$ motion planning benchmarks show that the average path costs in these environments approaches the costs of paths in k−PRM*. The resulting roadmap is significantly smaller both in terms of number of edges and nodes compared to roadmaps returned by k−PRM* and IRS that provide similar path quality. This allows fast online query resolution. The computational cost of constructing the roadmap is also evaluated.

II. Foundations

The C-space abstraction casts a robot's position and orientation as a point in a d-dimensional space. This paper considers the C-spaces of planar and rigid body configurations ($SE(2)$ and $SE(3)$), but IRS2 is applicable to any C-spaces with available metric and sampling functions.

Definition 1 (The Path Planning Problem): Given the set of free configurations $C_{\text{free}} \subset C$, initial and goal points $q_{\text{init}}, q_{\text{goal}} \in C_{\text{free}}$, find a continuous curve $\sigma \in \Sigma = \{\rho | \rho : [0,1] \rightarrow C_{\text{free}}\}$, $\sigma(0) = q_{\text{init}}$ and $\sigma(1) = q_{\text{goal}}$.

A. Variations of Roadmaps and Their Properties

PRM can solve the above problem by sampling configurations in C_{free} and connecting them with local planners [1]. During each iteration, the algorithm samples a point in C_{free} and adds it as a node to a roadmap. It then finds the neighbors within a δ-ball and tries to connect the sample to each neighbor. If the corresponding path exists in C_{free}, an edge between the two nodes is added. The roadmap eventually reflects the connectivity of C_{free} and can be used to answer queries. The online phase requires adding q_{init} and q_{goal} to the roadmap in a similar way during the roadmap construction. Then, a discrete graph search is performed to find a path on the roadmap between two query nodes q_{init} and q_{goal}.

Many variations exist for sampling and connecting configurations [10], [11], [12], [13], [14], [15]. Some alternatives focus on returning small, connected roadmaps that provide coverage [10], [15], [16]. Visibility-based Roadmaps reject nodes if they are not needed for coverage or connectivity [10]. Finer-grained measures of a sample's contribution for coverage and path quality have also been proposed [17], [18]. The Useful Cycles idea tests unnecessary edges connectivity-wise for their "usefulness" in terms of path quality [19]. The Useful Cycles was combined with the Reachability Roadmap Method to construct graphs that cover 2D and 3D C-spaces and provide high clearance paths [20].

There are methods that aim towards containing paths in all homotopic classes [3]. Nevertheless, they build dense roadmaps and may require an expensive online smoothing operation to transform the path to an optimal one. An alternative for improving path quality is to merge solutions returned by multiple calls to sampling-based planners [5]. Optimal paths can be constructed for specific queries [21] but resulting path costs depend on the density of the roadmap. Graph search algorithms, such as A*, offer optimality guarantees up to the discretization resolution but their computational complexity grows exponentially with the dimensionality of the C-space [22]. A recent optimization approach for computing optimal trajectories utilizes importance sampling [23].

Roadmaps cannot be used when there is no solution to the two-point boundary value problem, but tree-based kinodynamic planners can deal with such problems [24], [25]. Furthermore, tree-based planners already return a sparse graph. Nevertheless, they do not provide the preprocessing properties of roadmaps for multiple queries. Tree-based planners can also benefit from an approximate roadmap that estimates distances between states and considers C-space obstacles [26], [27]. Anytime RRT [28] has been proposed as an approach to incrementally improve path quality.

B. Graph Spanners

The current work is related to graph spanners [29]:

Definition 2 (Graph Spanner): A subgraph $(V, E' \subset E)$ of a weighted graph (V, E) is a t-spanner if $\forall (v_1, v_2) \in V$, the shortest path between them in (V, E') is no longer than t times the shortest path in (V, E).

Parameter t is known as the *stretch factor* of the spanner. A number of algorithms use the implicit weights of a Euclidean metric space to speed up the process but operate on *complete* Euclidean graphs [30], [31]. Thus, they can't be used in C-spaces with obstacles because obstacles remove edges of the complete graph. A simple method that does not require a complete graph accepts only edges that provide shortcuts and achieves good space complexity by reducing the number of edges from a $O(|V|^2)$ to $O|V|)$ [32]. The quadratic number of shortest path queries puts the overall time complexity, however, into $O(|V|^3 \log |V|)$. Much work has been done to find algorithms that reduce this. Most perform clustering in a preprocessing step and one method achieves linear complexity to the number of edges [33].

This idea was utilized in previous work by the authors to construct spanners of k−PRM* [8]. This was extended so as to define an incremental process for constructing a spanner, a method referred as Incremental Roadmap Spanner (IRS) [9]. Relative to most graph spanners which are formulated to operate on an existing graph, IRS2 can incrementally construct an additive roadmap spanner in a continuous space. IRS2 returns smaller roadmaps than IRS and allows the definition of a stopping criterion.

III. Improved Asymptotic Near-Optimal Planning

This section describes how to construct small roadmaps with the desirable properties of coverage and connectivity, while returning good quality paths.

A. Roadmap Spanners

The standard formulation of graph spanners does not allow for disconnecting nodes from the graph unless $t = \infty$. This is problematic when constructing spanners for continuous spaces, because it would require infinitely many edges. Consider the space of collision-free configurations C_{free}. An implicit graph (V, E) can be defined by taking all the elements of C_{free} as nodes and all the collision free paths between them as edges. Then, a *roadmap* is a subgraph $(V' \subset V, E' \subset E)$ of this implicit, dense graph with the following properties:

- All nodes in the dense graph are adjacent, connected by an edge, to a node that is also in the roadmap (coverage).
- The graph has one connected component for each component of the configuration space (connectivity).

The first property allows arbitrary query points to connect to the roadmap. The second guarantees that a path exists on the roadmap between any start and goal if a path between these nodes exists in the C-space. Using this formulation, a *roadmap spanner* can be defined as an implicit representation of an actual, infinite spanner over C_{free}. The implicit nodes of the spanner $V'_{imp} = V - V'$ must be able to connect to the explicit nodes V' by means of the implicit edges $E'_{imp} = E \cap (V_{imp} \times V')$. Furthermore, a roadmap spanner must guarantee that all shortest paths in the dense graph (V, E) are no longer than t times the corresponding shortest paths in the implicit graph $(V' \cup V'_{imp} = V, E' \cup E'_{imp})$ representing C_{free}.

Finding such a roadmap spanner with $t = 1$ corresponds to the standard Piano Mover's Problem, which is known to be P-SPACE hard in the general case. Sampling-based algorithms can solve such problems practically by relaxing the requirements of completeness to probabilistic completeness and optimality to asymptotic optimality. When $t > 1$, this must be further reduced to asymptotic *near*-optimality.

B. Description of IRS2

The general idea is to maintain a dense graph that better approximates the actual C_{free} as new samples are added. This graph is used to ensure that the sparse graph, which is the roadmap to be returned, contains enough nodes and edges to provide low local path costs.

a) IRS2*:* Algorithm 1 builds a dense graph (V, E) in the same way that PRM builds a roadmap. At every iteration, a point v is sampled from C_{free} (line 5). For each sample already in V within δ distance, an attempt is made to connect them using a local planner (line 6). This sample and any successful connections are always added to the dense graph (lines 7-8). The rest of the algorithm uses this dense graph to build the sparse roadmap (V', E'). First, edges that lead from the candidate sparse node v' to nodes already in the roadmap u' are gathered (line 12). Note that, because (V', E') is a subgraph of (V, E), no additional nearest-neighbor or collision work must be done. Information about the local C-space is used to determine if the sample should be added (line 13). If it is useful, the sample is added to V' (line **??**) along with any *useful* edges (line 14). Any neighbor of the sample that is not already in the roadmap is also consider for addition (line 15).

Algorithm 1 IncrementalRoadmapSpanner2(t, δ, M)

```
1:  V, E, V', E' ← ∅
2:  Q ← EmptyQueue
3:  m ← 0
4:  repeat
5:    v ← Sample(C_free)
6:    N_v ← CollisionFreeNeighbors(V, v, δ)
7:    V ← V ∪ {v}
8:    E ← E ∪ {(v, u) : u ∈ N_v}
9:    Enque(Q, v)
10:   while not Empty(Q) do
11:     v' ← Dequeue(Q)
12:     E'_v ← {(v', u') : u' ∈ N_v ∩ V'}
13:     if UsefulNode?(V', E', t, E'_v, v') then
14:       AddUsefulEdges(V', E', E'_v, t)
15:       Enqueue(Q, N_v − V')
16:   if v ∈ V' then
17:     m ← 0
18:   else
19:     m ← m + 1
20: until m = M
21: return (V', E')
```

b) UsefulNode: Algorithm 2 determines if it is necessary to add a sample to achieve coverage or maintain low path costs. Line 1 checks if the sample has any neighbors already in the roadmap. If it does not, it is a guard and must be added (line 2). Alternatively, the algorithm checks if the node is useful for path quality. The sample is needed if it connects two existing sparse nodes with a t times better path. To make this test, line 5 loops through the Cartesian product of all edges from the candidate sample v' to sparse nodes. For each pair of roadmap neighbors u'_1 and u'_2, the approach computes the least cost path from u'_1 to u'_2 through the sparse graph (line 6). If this path is greater that t times the cost of going from u'_1 to u'_2 through v', then v' is useful and is added (line 9) together with the edges from v' to u'_1 and u'_2 (line 10).

Algorithm 2 USEFULNODE?(V', E', t, E'_v, v')

```
1: if E'_v = ∅ then
2:     V' ← V' ∪ {v'}
3:     return true
4: else
5:     for all (v', u'_1), (v', u'_2) ∈ E'_v × E'_v do
6:         c ← SHORTESTPATH(V', E', u'_1, u'_2)
7:         c* ← d(v', u'_1) + d(v', u'_2)
8:         if c > t · c* then
9:             V' ← V' ∪ {v'}
10:            E' ← E' ∪ {(v', u'_1), (v', u'_2)}
11:            return true
12: return false
```

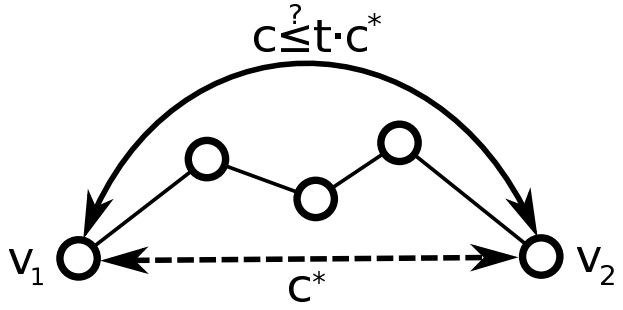

Fig. 2. An edge is not added if a path from its source to its target that less than t times the edge cost.

c) ADDUSEFULEDGES: Algorithm 3 is similar to the edge filter used in the Useful Cycles approach [19]. An edge is not added if there is already a path from its source to its target with a cost less than t times the edge cost (Figure 2).

The naive approach (lines 2-4) carries a large computational cost because of the graph search call. The actual cost of this path is not required, however. Only whether the path is longer than a certain amount. Thus, an optimization is to use a length-limited A* search, which does not expand nodes outside of the ellipse with foci v and u and major diameter $d(v, u)$. If a node outside this ellipse needs to be expanded, then the search can be stopped early and the edge (v, u) can be added to the roadmap. Another optimization is to maintain an incremental connected components data structure. If querying this data structure reveals that v and u are in different components, the edge can be added without graph search.

Algorithm 3 ADDUSEFULEDGES(V', E', E'_v, t)

```
1: for all (v, u) ∈ E'_v do
2:     c ← SHORTESTPATH(V', E', v, u)
3:     c* ← d(v, u)
4:     if c > t · c* then
5:         E' ← E' ∪ {(v, u)}
```

d) STOPPINGCRITERION: A traditional stopping criterion may be used, such as reaching time or space limits or when a solution that is sufficient for a specific task has been found. `IRS2` allows a more meaningful stopping criterion because of the concept of a failed sample. A failed sample is one that is not added to the roadmap and so does not improve solution quality or coverage. As the ratio of failed to successful nodes rises, the expected success of future samples is reduced. If samples are selected uniformly from the configuration space then this property holds globally, as with Visibility roadmaps [10]. In Algorithm 1, lines 3, 17, and 19 keep track of the number of consecutive failures m. If m exceeds a user defined value, M, then the algorithm is terminated.

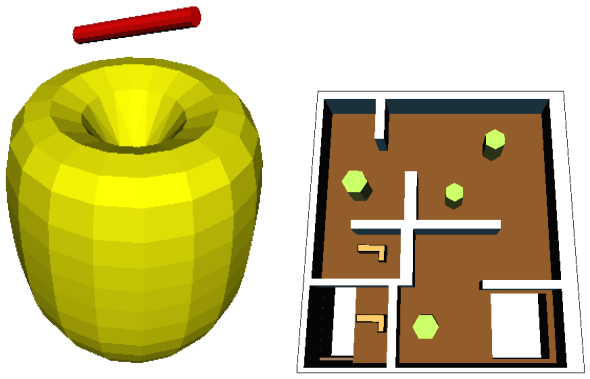

Fig. 3. Environments used in the $SE(3)$ experiments with example configurations for the robot (left: bug trap, right: cubicles). The environment shown in Figure 1 was used for the $SE(2)$ exeperiments.

C. Low Local Path Costs

While the global path cost properties of roadmaps produced by `IRS2` are difficult to reason about, the node filter described by Algorithm 2 will not reject nodes required to provide low local path costs. Similarly, the edge filter described in Algorithm 3 will not reject edges required for this property. The parameter δ defines the radius around which this property holds.

D. Differences from `IRS`

There are three differences between `IRS` and `IRS2`. The main improvement of `IRS2` over `IRS` is that not all samples are added to the roadmap. A procedure is used in the algorithm that evaluates the need to add samples in relation to coverage, connectivity and path quality.

Furthermore, a stopping criterion was not specified in the description of `IRS`. Here, it is explicitly related to the number of consecutive samples that are not considered useful. This is a natural method for determining how likely is the event that additional computation will improve the roadmap.

Finally, a fixed radius δ ball is used in `IRS2` to determine the number of neighbors with which to connect. This parameter represents the amount of solution degradation incurred by allowing samples to be dropped. In `IRS` the minimum radius required for asymptotic optimality is calculated for each sample. Providing guaranteed bounds on this additional degradation is the subject of future work.

IV. Experiments

A. Setup

Three algorithms were tested in the experiments: k−PRM*, IRS (stretch factors: $t = 1.5, 2.0, 3.0$), and IRS2 ($t = 1.5, 2.0, 3.0$ and $\delta = 0.1$ or 0.2 times the environment's extent). All runs used the Open Motion Planning Library (OMPL) [34] on 2GHz processors with 4GB of memory. Three environments from the OMPL library were chosen (Figures 1 and 3). For each setup, 10 roadmaps were constructed (10 minutes construction time) and 100 random query pairs were generated. Costs for each query are measured across the roadmaps and normalized relative to the smallest cost found in any experiment.

B. Path Cost and Space Requirements

Roadmap size was measured in terms of the amount of memory used. Each node requires several bytes for each configuration and book-keeping data related to edge information. Each edge requires only a few bytes for its cost and indices to the nodes it connects. IRS2 produces very small roadmaps while maintaining path costs comparable to the much larger roadmaps of the other algorithms (Figure 4). The results would be more dramatic if the space needed for each edge was not as low since the number of edges produced by k−PRM* is considerably higher.

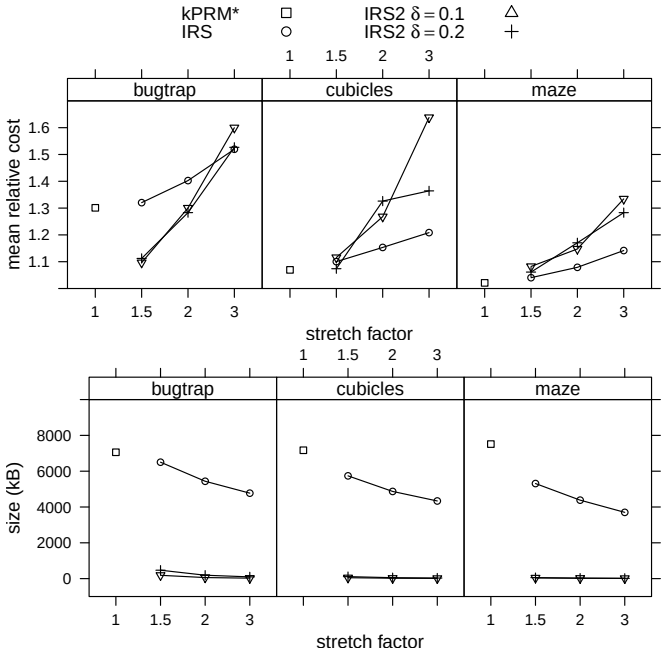

Fig. 4. In all environments IRS2 solved all query pairs and was smaller in size compared to the alternatives. k−PRM* and IRS found roadmaps with slightly lower cost solutions at the expense of many megabytes of memory.

C. Path Cost and Construction Time

Although the extra graph search operations that IRS2 must perform to filter nodes and edges have a significant overhead in asymptotic time complexity, experimental results suggest that it converges at a similar rate as k−PRM* and IRS (Figure 5).

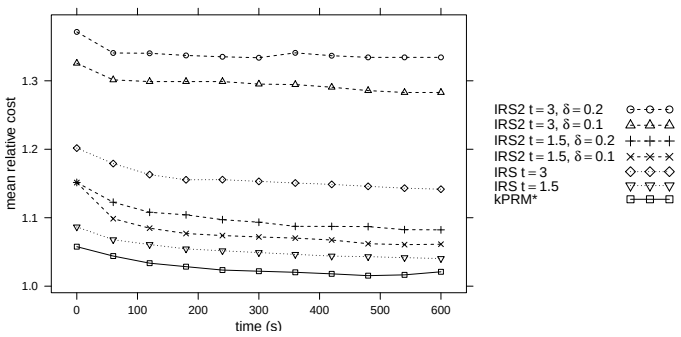

Fig. 5. Mean relative path cost as a function of construction time (maze).

D. Performance of the Stopping Criterion

The algorithms in these experiments were allowed to run for 10 minutes. If, instead, the more meaningful stopping criterion available to IRS2 was used, Figures 6 and 7 show what the mean solution cost and roadmap size would be.

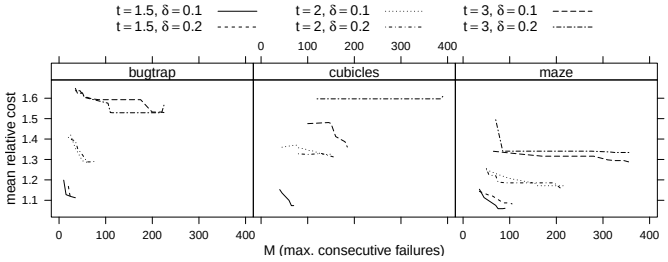

Fig. 6. Mean path cost as a function of parameter M.

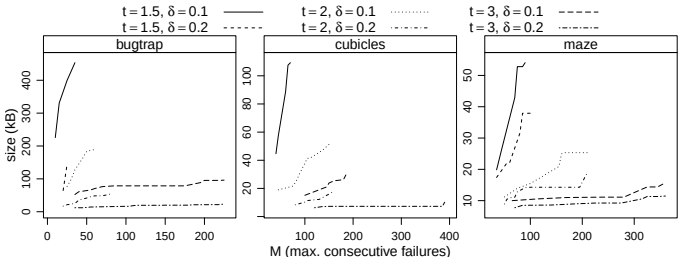

Fig. 7. Mean roadmap size as a function of the stopping criterion parameter M for each environment.

E. Effects of Smoothing

For applications where more time is available, the path cost can be improved by a short-cutting smoothing algorithm. Table I shows the mean improvement to solution path costs and smoothing time in the maze environment. More complex environments will require more time for smoothing so smoothing would be undesirable.

V. Discussion

An algorithm, IRS2, has been presented that produces very small roadmaps, which answer motion planning queries that converge to solution costs close to those of algorithms with path cost guarantees. Future experiments will

		cost		time (s)	
	δ	0.1	0.2	0.1	0.2
t	1.5	92%	90%	0.29	0.24
	2	88%	87%	0.28	0.22
	3	83%	83%	0.20	0.16

TABLE I

SMOOTHING COST SAVINGS AND TIME

be attempted in higher dimensional C-spaces, to include articulated systems with many degrees of freedom. The experiments presented in the work use the distance between the Euclidean projections of configurations as the cost function. A more general cost function would work as long as it forms a metric space in the C-space.

There are two limitations with the current work. There is not a proof of asymptotic near-optimality for the proposed algorithm even though experimentally the method returns good quality paths. Furthermore, the number of nodes that `IRS2` may add to the roadmap is infinite. It is interesting to investigate methods that generate roadmaps, which have asymptotic near-optimality guarantees, but also guarantee a finite number of nodes.

REFERENCES

[1] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation (TRA)*, vol. 12, no. 4, pp. 566–580, 1996.

[2] O. Nechushtan, B. Raveh, and D. Halperin, "Sampling-Diagrams Automata: a Tool for Analyzing Path Quality in Tree Planners," in *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, Singapore, December 2010.

[3] L. Jaillet and T. Simeon, "Path Deformation Roadmaps," in *Workhop on the Algorithmic Foundations of Robotics (WAFR)*, New York City, NY, July 2006.

[4] E. Schmitzberger, J. L. Bouchet, M. Dufaut, D. Wolf, and R. Husson, "Capture of Homotopy Classes with Probabilistic Roadmap," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Switzerland, 2002, pp. 2317–2322.

[5] B. Raveh, A. Enosh, and D. Halperin, "A Little More, a Lot Better: Improving Path Quality by a Path-Merging Algorithm," *IEEE Transactions on Robotics*, vol. 27, no. 2, pp. 365–370, 2011.

[6] S. Karaman and E. Frazzoli, "Incremental Sampling-based Algorithms for Optimal Motion Planning," in *Robotics: Science and Systems (RSS)*, Zaragoza, Spain, 2010.

[7] ——, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Reseach (IJRR)*, vol. 30, no. 7, pp. 846–894, June 2011.

[8] J. D. Marble and K. E. Bekris, "Computing Spanners of Asymptotically Optimal Probabilistic Roadmaps," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, San Francisco, CA, September 2011.

[9] ——, "Asymptotically Near-Optimal is Good Enough for Motion Planning," in *International Symposium of Robotics Research (ISRR)*, Flagstaff, AZ, August 2011.

[10] T. Simeon, J.-P. Laumond, and C. Nissoux, "Visibility-based probabilistic roadmaps for motion planning," *Advanced Robotics Journal*, vol. 41, no. 6, pp. 477–494, 2000.

[11] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "OBPRM: An Obstacle-based PRM for 3D Workspaces," in *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 1998, pp. 155–168.

[12] L. J. Guibas, C. Holleman, and L. E. Kavraki, "A Probabilistic Roadmap Planner for Flexible Objects with a Workspace Medial-Axis-Based Sampling Approach," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, vol. 1, Oct. 1999, pp. 254–259.

[13] E. Plaku, K. E. Bekris, B. Y. Chen, A. M. Ladd, and L. E. Kavraki, "Sampling-Based Roadmap of Trees for Parallel Motion Planning," *IEEE Transactions on Robotics and Automation (TRA)*, vol. 21, no. 4, pp. 587–608, 2005.

[14] G. Sanchez and J.-C. Latombe, "A Single-Query, Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking," in *International Symposium of Robotics Research (ISRR)*, 2001, pp. 403–418.

[15] D. Xie, M. Morales, R. Pearce, S. Thomas, J.-L. Lien, and N. M. Amato, "Incremental Map Generation (IMG)," in *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, New York City, NY, July 2006.

[16] R. Geraerts and M. H. Overmars, "Creating Small Graphs for Solving Motion Planning Problems," in *IEEE International Conference on Methods and Models in Automation and Robotics (MMAR)*, 2005, pp. 531–536.

[17] M. A. Morales, R. Pearce, and N. M. Amato, "Metrics for Analyzing the Evolution of C-Space Models," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2006, pp. 1268–1273.

[18] R. Pearce, M. Morales, and N. Amato, "Structural Improvement Filtering Strategy for PRM," in *Robotics: Science and Systems (RSS)*, June 2008.

[19] D. Nieuwenhuisen and M. H. Overmars, "Using Cycles in Probabilistic Roadmap Graphs," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2004, pp. 446–452.

[20] R. Geraerts and M. H. Overmars, "Creating High-Quality Roadmaps for Motion Planning in Virtual Environments," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Beijing, China, October 2006, pp. 4355–4361.

[21] J. Kim, R. A. Pearce, and N. M. Amato, "Extracting Optimal Paths from Roadmaps for Motion Planning," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 2, Taipei, Taiwan, 14-19 September 2003, pp. 2424–2429.

[22] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime search in dynamic graphs," *Artificial Intelligence Journal*, vol. 172, no. 14, pp. 1613–1643, 2008.

[23] M. Kobilarov, "Cross-entropy randomized motion planning," in *Robotics: Science and Systems (RSS)*, Los Angeles, CA, June 2011.

[24] S. M. LaValle and J. J. Kuffner, "Randomized Kinodynamic Planning," *International Journal of Robotics Reseach (IJRR)*, vol. 20, no. 5, pp. 378–400, May 2001.

[25] D. Hsu, R. Kindel, J. C. Latombe, and S. Rock, "Randomized Kinodynamic Motion Planning with Moving Obstacles," *International Journal of Robotics Reseach (IJRR)*, vol. 21, no. 3, pp. 233–255, Mar. 2002.

[26] K. Bekris and L. Kavraki, "Informed and Probabilistically Complete Search for Motion Planning under Differential Constraints," in *First International Symposium on Search Techniques in Artificial Intelligence and Robotics (STAIR)*, Chicago, IL, July 2008.

[27] Y. Li and K. E. Bekris, "Learning Approximate Cost-to-Go Metrics To Improve Sampling-based Motion Planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 9-13 May 2011.

[28] D. Ferguson and A. Stentz, "Anytime RRTs," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Beijing, China, Oct. 2006, pp. 5369–5375.

[29] D. Peleg and A. Schäffer, "Graph Spanners," *Journal of Graph Theory*, vol. 13, no. 1, pp. 99–116, 1989.

[30] J. M. Keil, "Approximating the Complete Euclidean Graph," in *1st Scandinavian Workshop on Algorithm Theory*. London, UK: Springer-Verlag, 1988, pp. 208–213.

[31] J. Gao, L. J. Guibas, and A. Nguyen, "Deformable spanners and applications." *Computational Geometry: Theory and Applications*, vol. 35, no. 1-2, pp. 2–19, Aug. 2006.

[32] I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares, "On sparse spanners of weighted graphs," *Discrete and Computational Geometry*, vol. 9, no. 1, pp. 81–100, 1993.

[33] S. Baswana and S. Sen, "A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs," *Random Structures and Algorithms*, vol. 30, no. 4, pp. 532–563, July 2007.

[34] I. A. Sucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library (OMPL)," http://ompl.kavrakilab.org, 2010.